Ignition MES Suite

User Manual

inductive automation
800.266.7798
www.inductiveautomation.com

MES User Manual

© Inductive Automation

All rights reserved. No parts of this work may be reproduced in any form or by any means - graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems - without the written permission of the publisher.

All Inductive Automation products are trademarks or registered trademarks of Inductive Automation. Other brand and product names are trademarks or registered trademarks of their respective holders.

Every effort has been made to make this book as complete and as accurate as possible, but no warranty or fitness is implied. The information provided is on an "as is" basis. Inductive Automation shall have neither liability nor responsibility to any person or entity with respect to any loss or damages arising from the information contained in this book.

Printed: November 2013 in the U.S.A.

Table of Contents

Part I Introduction 12

1 Production Model ... 12
2 Getting Help ... 13
3 Licensing and Activation ... 14

Part II OEE Downtime 18

1 Introduction ... 18
- OEE ... 18
- TEEP ... 20
- Production Count Tracking ... 20
- Down Time Tracking ... 20
- Production Scheduling ... 21

2 Getting Started ... 21
- Installation ... 21
  - Installing Modules ... 21
  - Configure Database ... 23
  - MES Module Settings ... 23
  - Demo Installation ... 24
- User Interface ... 25
  - Work Orders ... 26
  - Product Codes ... 27
  - Production Schedule ... 27
  - Operator Screen ... 29
  - Line Charts ... 32
  - Analysis Screen ... 33
  - Report Screen ... 37
- Dashboard ... 37
  - Area Summary ... 37
  - Enterprise Summary ... 39
  - Impromptu Analysis ... 40
  - Line Summary ... 40
  - Site Summary ... 41
  - TEEP ... 42
- Production Model ... 43
  - Production Item Settings ... 43
  - Adding Production Items ... 47
- Configuration ... 47
  - Components ... 48
  - Creating a Screen ... 48

3 Configuration ... 52
- MES Module Configuration ... 53
  - Datasource Settings ... 53
- Production Model Configuration ... 54
  - Production Model ... 55
    - Enterprise Configuration ... 55

© Inductive Automation

Site Configuration ... 57
Area Configuration ... 59
Line Configuration ... 61
Cell Group Configuration ... 70
Cell Configuration ... 72
Workday Routines ... 76
Downtime Reasons ... 78
Adding a Downtime Reasons ... 80
Editing a Downtime Reasons ... 82
Deleting a Downtime Reasons ... 82
Import / Export ... 82
Product Infeed ... 83
Adding a Product Infeed ... 83
Editing a Product Infeed ... 84
Deleting a Product Infeed ... 84
Import / Export ... 84
Product Outfeed ... 85
Adding a Product Outfeed ... 85
Editing a Product Outfeed ... 86
Deleting a Product Outfeed ... 86
Import / Export ... 86
Product Waste ... 87
Adding a Product Waste Counter ... 88
Editing a Product Waste Counter ... 88
Deleting a Product Waste Counter ... 89
Import / Export ... 89
4 Component Reference ... 89
Production Components ... 89
Production Line Selector ... 89
Production Cell Selector ... 91
Product Code Selector ... 92
Product Code Table ... 93
Product Code Line Table ... 94
Product Code Properties Table ... 95
Production Comments Panel ... 96
Product Code Controller ... 99
Analysis Controller ... 100
Production Analysis Selector ... 106
Production Stored Analysis Selector ... 110
Production Bar Chart ... 111
Production Pie Chart ... 112
Analysis Table ... 114
Down Time Components ... 116
Down Time Table ... 116
Performance Indicator ... 125
Line Run Selector ... 126
Analysis Time Chart ... 128
Schedule Components ... 131
Work Order Selector ... 131
Work Order Table ... 132
Work Order Controller ... 133
Line Schedule Selector ... 135
Schedule Day View ... 137
Schedule Week View ... 139

Schedule Month View ... 141
Schedule Date Selector ... 143
Schedule Entry Controller ... 144
Schedule Controller ... 149
Time Selector ... 152
Line Schedule View ... 153
5 Production OPC Values ... 162
Using OPC Values ... 163
OPC Value Reference ... 164
Project ... 164
Enterprise ... 165
Site ... 166
Area ... 167
Line ... 169
Cell ... 176
Additional Factors ... 180
Workday Routine ... 181
Downtime Reasons ... 182
6 Binding Function Reference ... 183
Analysis ... 185
Analysis Filter ... 185
History ... 186
Downtime History ... 186
Production History ... 187
Scheduled vs. Actual ... 188
7 Scripting ... 189
Line Events ... 190
Custom OEE Calculations ... 191
OEE Factor Cap ... 192
Gateway Scripts ... 193
Client/Designer Scripts ... 203
8 Analysis Providers ... 211
Comment ... 212
Downtime ... 213
OEE ... 215
Schedule ... 218
TEEP ... 219
9 Miscellaneous ... 221
Additional Factors ... 221
Production Rate Calculation ... 221
Part III SPC Quality 224
1 Introduction ... 224
SPC Charts ... 224
Scheduling Samples ... 227
Evaluating Signals ... 227
2 Getting Started ... 227
Installation ... 227
Existing Ignition System ... 227
Installing Modules ... 227
New Ignition System ... 229

© Inductive Automation

Selecting Install Options ... 229
Configure Database ... 229
MES Module Settings ... 230
Demo Installation ... 231
User Interface ... 232
Quality User Screens ... 233
Overview ... 234
Sample Definitions ... 235
Sample Entry ... 241
SPC Control Charts ... 243
Control Charts ... 244
Analysis ... 247
3 Configuration ... 250
MES Module Configuration ... 250
Datasource Settings ... 251
Production Model Configuration ... 252
Production Module ... 252
Enterprise Configuration ... 253
Site Configuration ... 255
Area Configuration ... 257
Line Configuration ... 258
Location Configuration ... 259
Control Limits ... 262
Overview ... 263
Default Control Limits ... 263
Add Control Limits ... 263
Edit Control Limits ... 267
Delete Control Limits ... 267
Import/Export ... 267
Out of Control Signals ... 268
Overview ... 268
Default Signals ... 269
Add Signals ... 269
Edit Signals ... 272
Delete Signals ... 272
Import/Export ... 272
Sample Intervals ... 273
Overview ... 273
Default Intervals ... 273
Add Intervals ... 273
Edit Intervals ... 275
Delete Intervals ... 275
Import/Export ... 275
SQLTag Sample Collectors ... 275
Overview ... 275
Add Sample Collectors ... 276
Edit Sample Collectors ... 277
Delete Sample Collectors ... 277
Import/Export ... 277
4 Component Reference ... 278
Quality Components ... 278
Definition List ... 278
Definition Attribute List ... 282

Definition Location List ... 284
Definition Control Limit List ... 286
Definition Signals List ... 287
Location Selector ... 287
Interval Selector ... 289
Datatype Selector ... 290
Definition Selector ... 291
Location Sample List ... 292
Sample Entry ... 298
SPC Components ... 301
SPC Selector ... 302
Stored SPC Selector ... 307
SPC Controller ... 309
Histogram Chart ... 313
Pareto Chart ... 316
Xbar and R Chart ... 320
Xbar and S Chart ... 326
Median and Range Chart ... 332
Individual and Range Chart ... 338
P-Chart ... 344
NP-Chart ... 350
U-Chart ... 355
C-Chart ... 361
5 Quality OPC Values ... 367
Using OPC Values_2 ... 368
SPC OPC Value Reference ... 369
Project ... 369
Enterprise ... 370
Control Limits ... 371
Signals ... 371
Intervals ... 372
Site ... 373
Area ... 374
Line ... 375
Location ... 378
Additional Factors ... 381
Tag Collectors ... 382
6 Scripting ... 384
Production Location Events ... 384
Before Sample Updated Event ... 384
After Sample Updated Event ... 384
Sample Approval Updated Event ... 385
Sample Coming Due Event ... 385
Sample Due Event ... 385
Sample Interval Event ... 386
Sample Overdue Event ... 388
Sample Waiting Approval Event ... 388
Signals Evaluated Event ... 388
Signal Evaluation Results ... 389
Signal Out of Control Event ... 389
Signal in Control Event ... 390
Sample Due State Types ... 390
Object Reference ... 390

Sample ... 390
Sample Data ... 394
Attribute Data Type ... 395
Sample Additional Factor ... 396
Sample Definition ... 397
Sample Definition Attribute ... 402
Sample Definition Location ... 405
Sample Definition Control Limit ... 407
Sample Definition Signal ... 408
Control Limit Kind Type ... 409
SPC Category Types ... 410
Signal Kind Types ... 410
Control Limit Calculated Value ... 411
Scripting Functions ... 411
sample.production ... 411
sample.production.utils ... 411
cancelLocationProductCode ... 412
setLocationProductCode ... 412
sample.quality ... 413
sample.quality.definition ... 413
getNew ... 413
getSampleDefinition ... 413
addSampleDefinition ... 414
updateSampleDefinition ... 415
sample.quality.sample.data ... 415
getNew ByDefUUID ... 415
getNew ByName ... 416
getCreateSampleByDefUUID ... 416
getCreateSampleByName ... 417
getSample ... 418
updateSample ... 418
approveSample ... 419
unapproveSample ... 419
removeSample ... 420
sample.quality.spc.controllimit ... 420
setControlLimitValue ... 420
calcControlLimitValue ... 421
7 Analysis Providers ... 423
Quality ... 423
Part IV Recipe / Changeover 428
1 Introduction ... 428
Recipe Types ... 428
Production Model ... 430
Default Values ... 433
Master Recipes ... 435
Sub Recipes ... 436
Editing Recipes ... 438
Recipe Change Log ... 441
Recipe Security ... 443
Recipe Scaling ... 444
Variance Monitoring ... 444
Selecting Recipes ... 448

Production OPC Server ... **451**
2 Installation ... **454**
Existing Ignition System ... **454**
Installing Modules ... 454
New Ignition System ... **456**
Selecting Install Options ... 456
Configure Database ... **456**
MES Module Settings ... **457**
3 Configuration ... **458**
MES Module Configuration ... **458**
Datasource Settings ... 458
Production Model Configuration ... **460**
Production Model ... 460
Enterprise Configuration ... 461
Site Configuration ... 462
Area Configuration ... 463
Line Configuration ... 464
Cell Configuration ... 466
Cell Group Configuration ... 467
Location Configuration ... 469
Sub Recipe Mask ... **470**
Sub Recipe Mask Setting ... 470
Recipe Values ... **470**
Adding a Recipe Value ... 471
Editing a Recipe Value ... 475
Deleting a Recipe Value ... 475
Import / Export ... 475
4 Component Reference ... **476**
Recipe Editor ... **476**
Recipe Changelog Viewer ... **487**
Recipe Variance Viewer ... **489**
Recipe Selector Combo ... **491**
Recipe Selector List ... **492**
5 Analysis Providers ... **493**
Recipe Analysis Provider ... **494**
Recipe Variance Analysis Provider ... **496**
Sub Product Code Variance Analysis Provider ... **498**
Recipe Change Log Analysis Provider ... **500**
6 Production OPC Values ... **502**
Enterprise ... **503**
Site ... **504**
Area ... **505**
Line ... **506**
Cell ... **508**
Cell Group ... **510**
Location ... **512**
7 Scripting ... **514**
Client / Gateway Scripts ... **514**
getDefaultValues ... 515
setPathDefaultValue ... 516
revertPathDefaultValue ... 517
createSubProductCode ... 518

renameSubProductCode ... 519
deleteSubProductCode ... 520
setPathRecipeValue ... 521
revertPathRecipeValues ... 522
createRecipe ... 523
renameRecipe ... 524
deleteRecipe ... 525
addItemToRecipe ... 526
removeItemFromRecipe ... 527
getChangelogHistory ... 528
getRecipeVariances ... 529
getItemRecipeList ... 530
getCurrentItemRecipe ... 531
getRecipeValues ... 532
isItemRecipeMonitoringEnabled ... 533
setItemRecipe ... 534
cancelItemRecipe ... 535
readItemCurrentValues ... 536
exportRecipe ... 537
importRecipe ... 538
Recipe Value Scripts ... 538
Evaluate Variance Script ... 538
Request Value Script ... 539
Object Reference ... 540
ChangelogFilters ... 540
VarianceFilters ... 542
ItemRecipeValue ... 543
RecipeTag ... 545

Part V Instrument Interface Module 548

1 Introduction ... 548

2 File Monitor Settings ... 549

3 Serial Settings ... 551

4 Parse Template ... 554

5 Component Reference ... 559

File Monitor ... 559
Serial Controller ... 564

6 Scripting ... 573

Object Reference ... 573
Parse Results ... 573
Parse Value ... 574
Parse Row Collection ... 575
Parse Row ... 576
Gateway Scripts ... 577
Client/Designer Scripts ... 577

Index 579

Introduction

Part I

1 Introduction

1.1 Production Model

To start out, it is important to define what the production model is, which is heavily referred to when dealing with OEE and downtime.
A production model defines your manufacturing or process in tree view form. It provides an organized way to easily configure, control and analyze your facility.

Production Model Tree

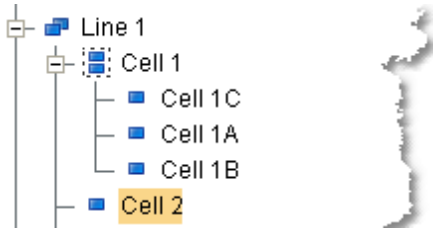
© Inductive Automation

Enterprise
The *enterprise* is the highest level of the production model and typically represents a manufacturing company. A company may have one or more production facilities.

Site
A *site* is a geographical production location and is part of an *enterprise*. Separating your *enterprise* into multiple production sites allows for comparing OEE, downtime and production information between them.

Area
An *area* is a physical or logical grouping of production *lines*.

Line
A *line* is a collection of one or more cells and/or cell groups that run a single product at any given time. Typically, the product flows from one cell or cell group to the next in sequence until the product, or sub assembly, being produced is complete.



Cell Group Tree

Location
A location is the space where a sample is collected. This can be placed under an area or a line and is used only in the SPC Quality Module

Cell Group
A *cell group* contains two or more *cells*. Typically, these *cells* occur at the same time in the sequence of the *line* instead of one after another, causing the *cell group* to act as a single sub process or step within the production.

Cell
The *cell* is a single machine, sub process or step required in the manufacture a product. The product may be a hard product such as used in packaging, adding liquid or powder, etc. Packaging machines are a common example, but a cell applies to processes also.

1.2 Getting Help

There are multiple methods of getting help with both Ignition and the MES modules:

Online Forum
From www.inductiveautomation.com website, the online support forum can be accessed to search for solutions and post questions. It is actively patrolled by Inductive Automation staff and many knowledgeable users.

Email Support
E-mail support is available at support@inductiveautomation.com

Phone Support
You can reach us during business hours 8am-5pm Pacific time at 1-800-266-7798. Support charges may apply. 24-hour support is also available for an addition fee.

© Inductive Automation

Design Services

Inductive Automation has design support staff skilled in working with IT, maintenance, production departments and integrating the OEE Downtime and Scheduling module to the plant floor and ERP systems. For more information, contact sales.

1.3 Licensing and Activation

Trial Mode

The OEE Downtime and Scheduling module follows the same trial operation as Ignition. The OEE Downtime and Scheduling module can be used for 2-hours at a time, with no other restrictions. At the end of the trial period, the system will stop logging data to the database, display expired trial overlays on live values and clients will see a demo screen. By logging into the gateway, you may re-start the demo period, and enable another 2 hours of execution. The demo period may be restarted any number of times.

You may install a unlicensed OEE Downtime and Scheduling module into a licensed Ignition server. The Ignition server licensing will not be affected and the OEE Downtime and Scheduling module will operate in Demo mode.

Licensing

The OEE Downtime and Scheduling license can be purchased along with, or separately from, the Ignition license. Despite the modular licensing, each Ignition server only has a single CD-Key and license file. That is, there is a single license file that dictates which modules are current activated.

When module(s) are purchased, you will receive a CD-Key - a six digit code that identifies your purchase. You then use this CD-Key to activate the software through the Ignition Gateway. Activation is a process by which the CD-Key and its associated parameters get locked to the machine that you are activating. If you adding an additional module, your account will be updated, and you can re-use your existing CD-Key to activate the new features. For this reason, if you purchased the OEE Downtime and Scheduling module separately from the Ignition server, the OEE Downtime and Scheduling license will have to be added to your existing CD-Key.

It is possible to inactivate your CD-Key, freeing it for activation on a different machine.

Not all production facilities have the large number of *lines* and *cells* while others do. For this reason there are two basic editions to choose from to meet your situation:

Standard License

The Standard edition provides OEE, downtime and scheduling functionality for unlimited production *areas*, *lines* and *cells*. Includes the OEE, downtime and schedule engine; configuration software; user interface screens; enhanced analysis tools; and reports. There are no restrictions on the number of tags, logged data items, screens or clients (users).

Line License

The Standard edition provides OEE, downtime and scheduling functionality for a single production *lines*. Multiple Line Licenses can be installed on a single server. There is no limit on the number of cells that a *line* can be configured for. Includes the OEE, downtime and schedule engine; configuration software; user interface screens; enhanced analysis tools; and reports. There are no restrictions on the number of tags, logged data items, screens or clients (users).

Enterprise Extension

In addition to the above editions, the Enterprise Extension allows analysis and reporting across multiple physical production *sites* from anywhere on your network. Compare efficiency and downtime by

© Inductive Automation

production *line*, operator, user defined values and more. Requirements: Standard or Line License for the OEE Downtime Module, and the Reporting module.

Activation

Activation, as mentioned above, is the method by which a cd-key is locked down to the install machine, and the modules are notified of their license state. It is a two step process that can be performed automatically over the internet, or manually through email or the Inductive Automation website.

Step 1 - Enter CD-Key

When the software is purchased, you are provided with a *six digit CD-key*. After logging into the gateway configuration, go to Licensing > Purchase or Activate, and select "Activate".
Enter your CD-key.

Step 2a - Activate over Internet

If your computer has internet access, activating over the internet is the easiest option. A secure file will be created with your cd-key, and sent to our servers. The response file will then be downloaded and installed, completing the entire process in seconds.

OR

Step 2b - Activate Manually

If you do not have internet access on the installation machine, you must activate manually. In this process, an activation request file is generated (activation_request.txt). You must then take this file to a machine with internet access, and email it to support@inductiveautomation.com, or visit our website to activate there. Either way will result in a license file (license.ipl) being generated, which you then must take back to the Gateway machine and enter into the License and Activation page.

© Inductive Automation

OEE Downtime

Part II

inductive automation
800.266.7798
www.inductiveautomation.com

2 OEE Downtime

2.1 Introduction

Improving production efficiency is the key to improving profit and reducing capital expenditures. It can make the difference competitively; however, it can also be very challenging because it requires more than just installing software. Improving efficiency requires commitment from management, maintenance, production and IT departments, as well as integration, training, actions to reduce downtime and new operational procedures. The OEE Downtime and Scheduling Module helps you to diagnose the inefficiencies within your production, allowing you to make improvements on the line and between employees.

The first step in improving efficiency is knowing where you are starting from. Think of it like improving the gas mileage of your car. You must start by determining your current gas mileage before you can begin making changes to improve your mileage. Once you know your existing OEE and have tracked the causes of downtime, then you can finish the process and start fixing the sources of your production inefficiencies.

But why combine OEE, downtime and scheduling into one module? The OEE Downtime and Scheduling Module does not require the use of all three functions, but we packaged them together because the combination provides the best tools for the improvement of production efficiency. If only downtime was tracked, then you would not see the full picture. Downtime only informs you as to whether or not a machine is running, not if the machine is actually producing a quality product. Or if only OEE was tracked, you would know that efficiency is lower than normal, but not why or what actions to take to improve it. Low efficiencies also result from ineffective procedures or a lack of communications between departments. This is where the scheduling helps by providing current schedule information to all associated departments, improving communication and reducing unnecessary delays. The OEE Downtime and Scheduling Module allows you to see the whole picture, resulting in the improvement of your production in every aspect.

2.1.1 OEE

OEE stands for Overall Equipment Effectiveness and is used to monitor manufacturing effectiveness. The resulting OEE number, represented as a percentage, is generic and allows comparisons across differing industries.

Efficiency is not simply the ratio of machine run time to scheduled time. Look at the situation of your manufacturing *line* or process running at half speed with 0 downtime. This is truly only 50% efficient. Or what if 10% of the product being produced does not meet your minimum quality and must be reworked. This equates to 90% efficient, which does not take into account the effort to rework or the losses of raw material.

© Inductive Automation

There are three factors, all represented as a percentage, taken into consideration for the final OEE result:

OEE Availability

OEE Availability is the ratio between the actual run time and the scheduled run time. The scheduled run time does not included breaks, lunches and other pre-arranged time a production *line* or process may be down.

Example:
If a line is scheduled for one 8 hour shift with two 15 minute breaks and one 30 minute lunch, then the scheduled time is 7 hours (determined from 8 hours - 15 minute break - 15 minute break - 30 minute lunch). If during the production run, there are 25 downtime events totaling to 45 minutes of downtime, then the run time is 6 hours and 15 minutes (derived from 7 hours of scheduled time - 45 minutes). The OEE Availability of 89% is calculated by actual run time divided by scheduled run time, or 6 hours 15 minutes divided by 7 hours.

OEE Performance

OEE Performance is the ratio between the actual number of units produced and the number of units that theoretically can be produced based on the standard rate. The standard rate is rate the equipment is designed for.

Example:
If a work cell is designed to produce 10 units per minute we can calculate the theoretical amount of units it can produce in a given amount of time. Using the 6 hours and 15 minutes of actual run time from the above example, a total of 3750 units would be produced. Calculated by taking 6 hours and 15 minutes (375 minutes) times 10 units per minute. If the actual number of units produced is 3000, then the OEE Performance is 80% (calculated by 3000 / 3750).

OEE Quality

OEE Quality is the ratio between good units produced and the total units that were started.

Example:
Taking the number of units produced from above of 3000, if 200 units were rejected at the quality inspection station, then 2800 good units are produced . The OEE Quality is 93% calculated from 2800 divided by 3000.

The final calculation is OEE = Availability x Performance x Quality.

Example:
Using all the numbers from above, 89% x 80% x 93% = 66%.

This may seem like a low number but it is important to kept in mind that the OEE is not to be compared to 100%. The OEE result from this production run is compared to other production runs; however, using Inductive Automation's OEE Downtime and Scheduling module allows much more than just comparing OEE results between production runs. It allows you to compare OEE results between operators, viscosity, mechanics, products, raw material vendors and any user defined factor you can think of.

© Inductive Automation

2.1.2 TEEP

Where OEE represents the equipment efficiency during a production run, Total Effective Equipment Performance (TEEP) represents the equipment utilization against a calendar period. For example, 365 days a year, or 24 hours a day. It can also be thought of as asset utilization and will help in the decision making process of purchasing new equipment.

There are two factors used to calculate TEEP:

Loading

Loading is the ratio between the scheduled time for the production *line* (or process) and the calendar time.

Example:
If a production line is scheduled for 5 days, 24 hours each day, over a 7 day period, then the loading is 71% calculated by (5 x 24) / (7 x 24).

OEE

OEE = Availability x Performance x Quality as described in the previous section.

The calculation is TEEP = Loading * OEE

Example:
To simplify this example we will use made up OEE result of 82%. The actual OEE value used must be the OEE result for all production runs of the same calendar time period that were used to calculate the Loading value.
*TEEP = 71% * 82%*
The TEEP result is 58% .

2.1.3 Production Count Tracking

For OEE calculations to be performed, production count information is required. At a minimum, the outfeed production count for a production *line* is needed. Additional production count information can be configured, leading to more OEE calculations.

For example, if the infeed production count is configured for a production, then product accumulation and waste can be calculated.

2.1.4 Down Time Tracking

OEE provides a method to monitor the efficiency of your production facility and tracking downtime provides information of where to focus efforts to improve efficiency. Think of it this way, if your production *line* typically runs at 69% OEE, what actions do you take to increase it? OEE alone doesn't tell you what factors are preventing your efficiency from being higher than 69%.

In the simplest form, downtime tracking will identify the production cell (machine or process) that is preventing you production *line* from producing product. This can be done manually, but history has shown that manually collected downtime information is not accurate. In addition, if it is manually collected on paper log sheets, then someone has to further enter the details into a program or spreadsheet to be able to organize it into actionable information used to focus your efforts to make improvements. Putting recording inaccuracies, extra labor and typos aside, by the time the information is available, it is old.

Tracking downtime automatically or semi-automatically solves the issues associated with manual

© Inductive Automation

tracking. In a perfect world, monitoring all downtime reasons automatically is the ideal solution. But in the real world, this can be difficult, pricey, or just not practical. For this reason, it is important for downtime tracking software to support an automatic reason detection with a manual override. For example: if an operator presses the stop button because they see a bottle laying on its side feeding into a filler, then the only automatic reason that can be detected is "operator pressed stop button". Now the operator should be able to override this reason with more specific information.

Once the period of time that production cells were not producing product and the associated reasons are recorded, analyzing the summary of the reasons will identify where effort should be focused to improve efficiency.

2.1.5 Production Scheduling

A lot of coordination must be used when scheduling production. If one item is not in unison with the rest, then production *line* efficiencies will drop. If raw material is not at the *line* when the *line* is ready to start production, then *line* production is waiting. Even if this is just 10 minutes, it negatively reduces the production *line* efficiency.

In some operations, production schedules change, sometimes at the last minute, making if difficult and forcing employees to rely on a verbal updates involving multiple people. It becomes an issue of how much effort is being consumed to do so and how many times are there hiccups.

By instantaneously propagating schedule changes to all departments, combined with tools to track required, scheduled, produced and remaining production information, can help make an operation run smoother.

2.2 Getting Started

This getting started guide will step you though OEE Downtime and Scheduling module installation, demo installation, the demo user interface and configuration features.

2.2.1 Installation

Follow the next four sections to install the complete OEE Downtime and Scheduling. If you just want in install the core modules and skip the demo, follow the next three sections.

2.2.1.1 Installing Modules

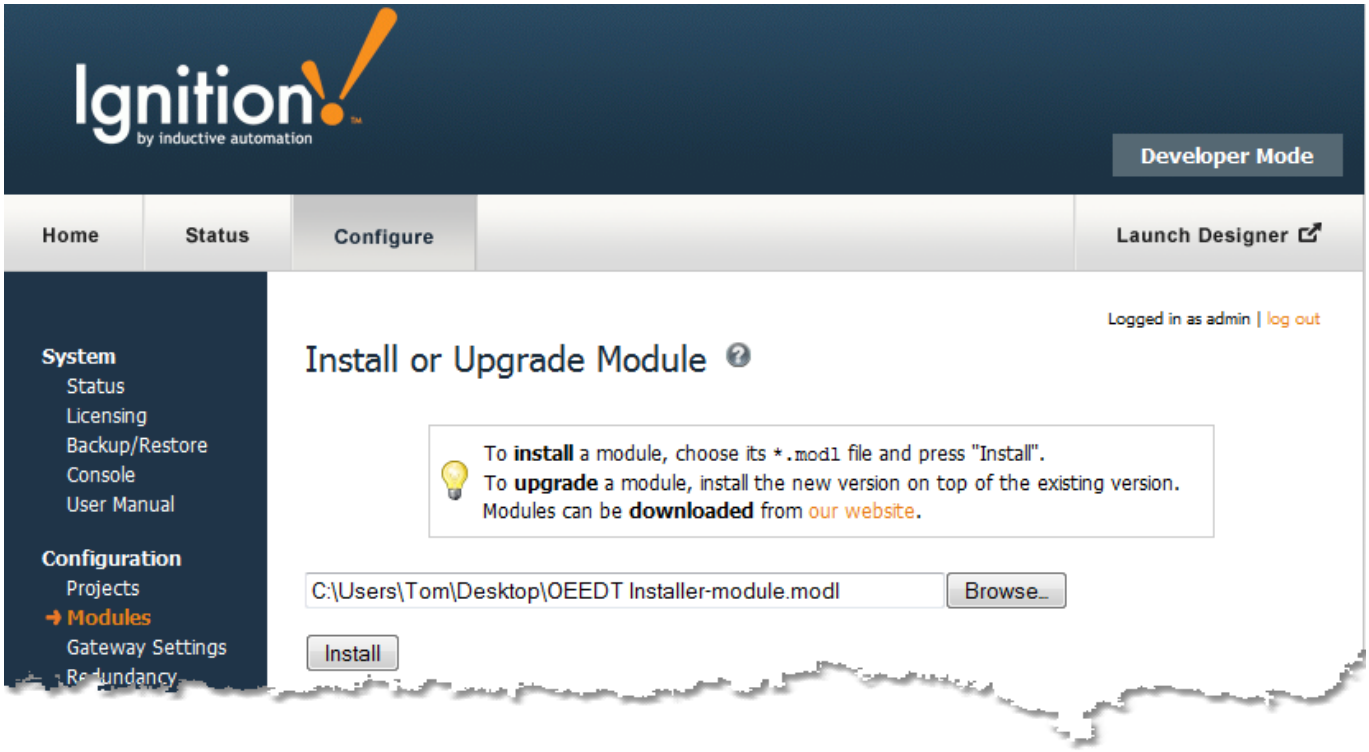
To install the OEE Downtime and Scheduling module on to an existing Ignition server, follow the steps below:

Before installing the OEE Downtime and Scheduling module, it is recommended to first setup the database connection that will be used to store OEE, downtime and scheduling data.

1. **Download the OEEDT Installer.modl module**
   from the Inductive Automation download website. It will be under the MES modules heading.

2. **Install the OEEDT Installer.modl module**
   Navigate to the Modules page of the Ignition gateway. At the bottom of the list of already installed modules, click the Install or Upgrade a Module... link. Next, browse to the OEEDT Installer.modl file and click the install button as shown below.

© Inductive Automation



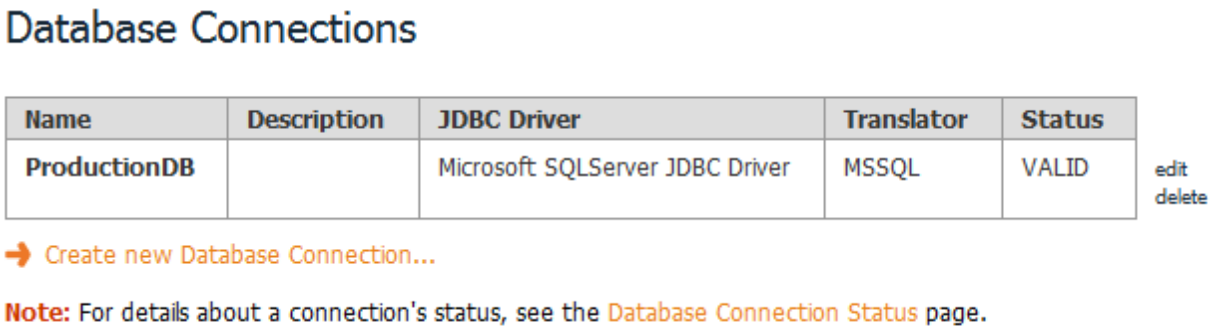
Install Ignition Module

© Inductive Automation

The OEEDT Installer module will install all required modules. These are the Production, OEE Downtime and Scheduling modules. It is important to keep in mind not to install or update these module individually. Instead, it should be done by updating the OEEDT Installer module.

2.2.1.2 Configure Database

OEE, downtime and schedule data is stored in databases external to Ignition. These database(s) are setup in the gateway configuration section by selecting the **Databases> Connections** section from the left-hand configuration menu. See the Ignition documentation for more information on setting up a database connection. Below shows a typical database connection that is required for the OEE, Downtime and Scheduling module.



Sample Database Connection

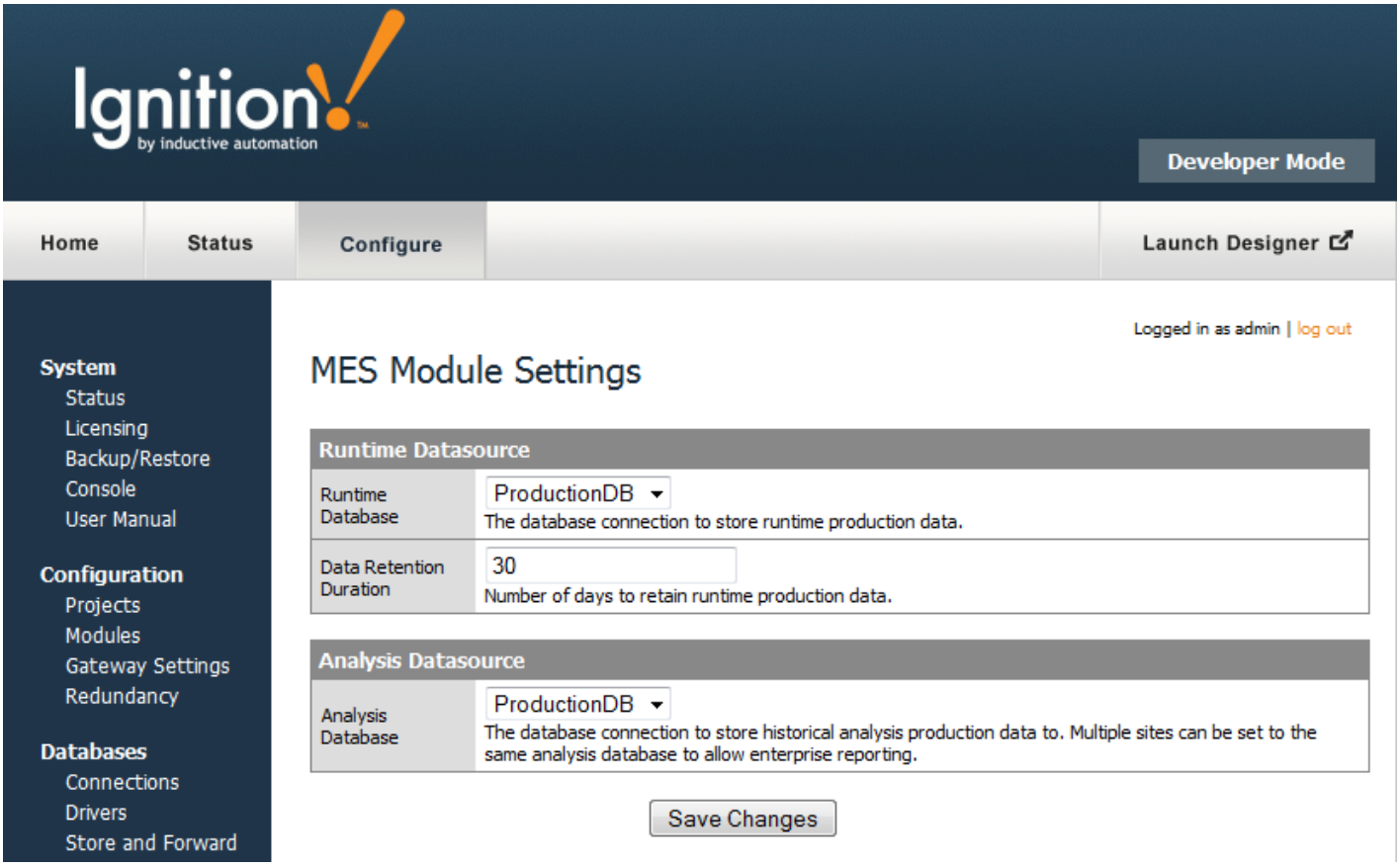
2.2.1.3 MES Module Settings

The OEE Downtime and Scheduling module stores data in a SQL database. Because Ignition can be configured to multiple databases, the MES Module Settings configuration page is used to select which databases to store OEE, downtime and scheduling data. If only one database has been configured in Ignition, then it will be selected by default.

To change the MES module settings, go to the configuration section in the gateway and select the **MES Modules> Settings** section from the left-hand side configuration menu.

Once a database connection is created, and if only one database connection exists, then it will be automatically selected to be used by the MES modules.
If more than one database connection exists, then the desired database connection can be selected to be used by the MES modules as shown below.

© Inductive Automation



MES Module Settings Page

For more information on the MES Module Settings, see MES Module Configuration.

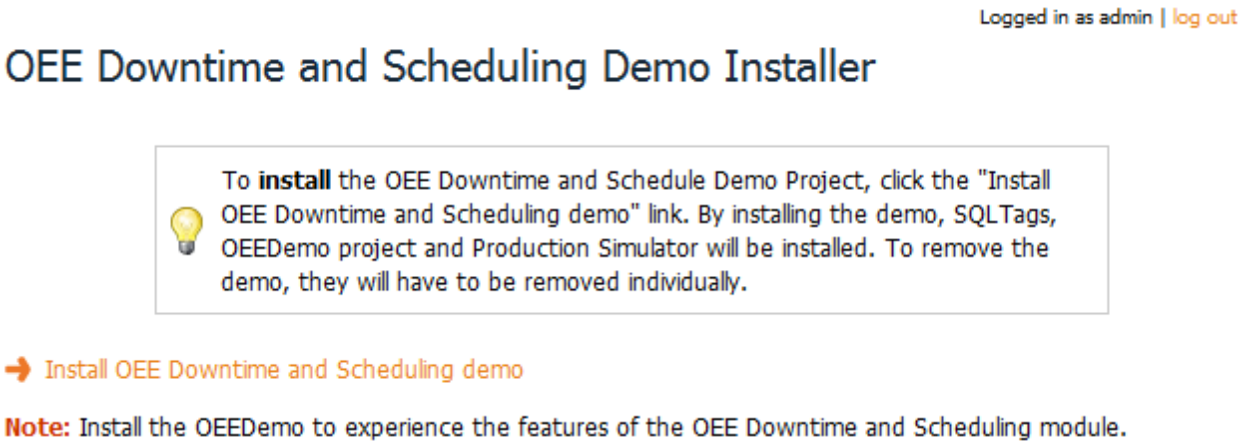
2.2.1.4 Demo Installation

The OEEDemo project can be used to quickly start using and evaluate the features of the OEE Downtime and Scheduling module.

By installing the OEEDemo, SQLTags will be imported, the Production Simulator will be installed, the demo Ignition project will be installed and sample database data will be added. To remove the demo, each of these components will have to be manually removed.

To install the OEE Demo project, go to the configuration section in the gateway and select the **MES Modules> Demo** section from the left-hand side configuration menu. Next, click on the → Install OEE Downtime and Scheduling demo link. This will take a few seconds as it is installing and making all of the configuration changes to run the OEE Demo.

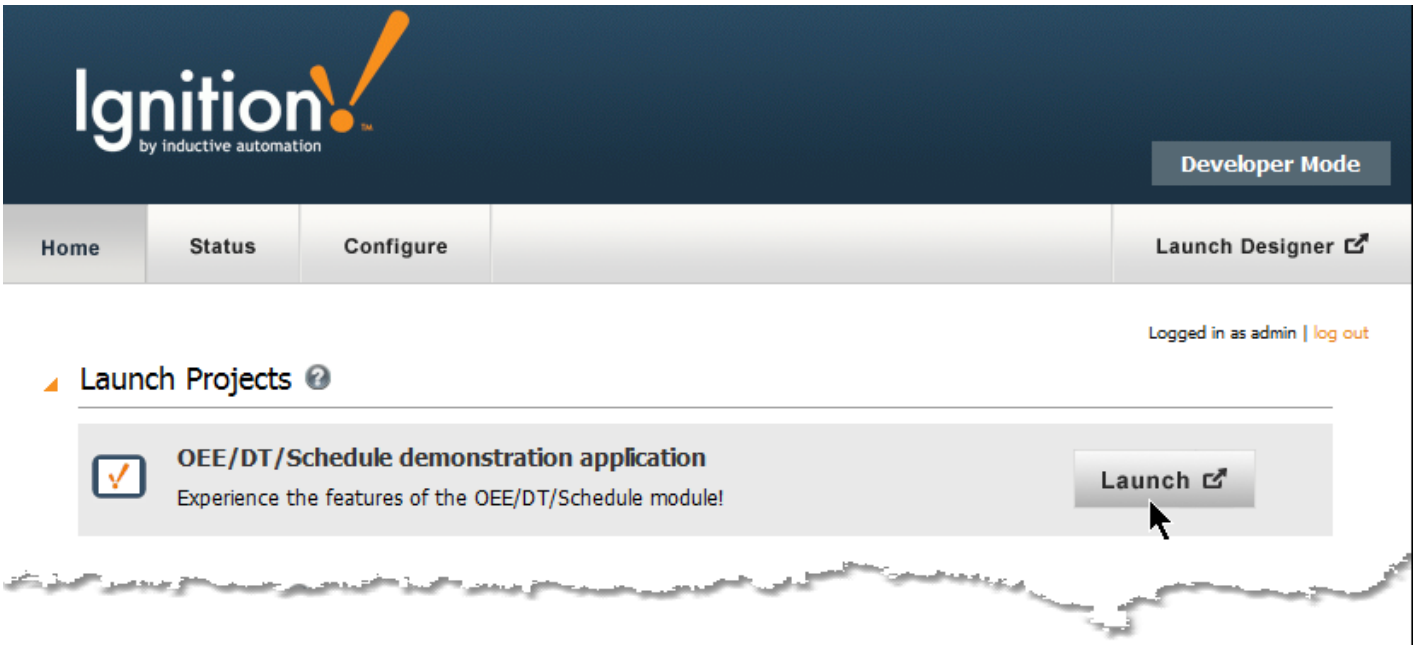
If the OEE Demo has already been installed, then there will be a note stating so. If a database connection has not been installed, a note will appear stating a database connection is needed before installing the demo.


Logged in as admin | log out
OEE Downtime and Scheduling Demo Installer
To install the OEE Downtime and Schedule Demo Project, click the "Install OEE Downtime and Scheduling demo" link. By installing the demo, SQLTags, OEEDemo project and Production Simulator will be installed. To remove the demo, they will have to be removed individually.
Install OEE Downtime and Scheduling demo
Note: Install the OEEDemo to experience the features of the OEE Downtime and Scheduling module.

Demo Installation Page

2.2.2 User Interface

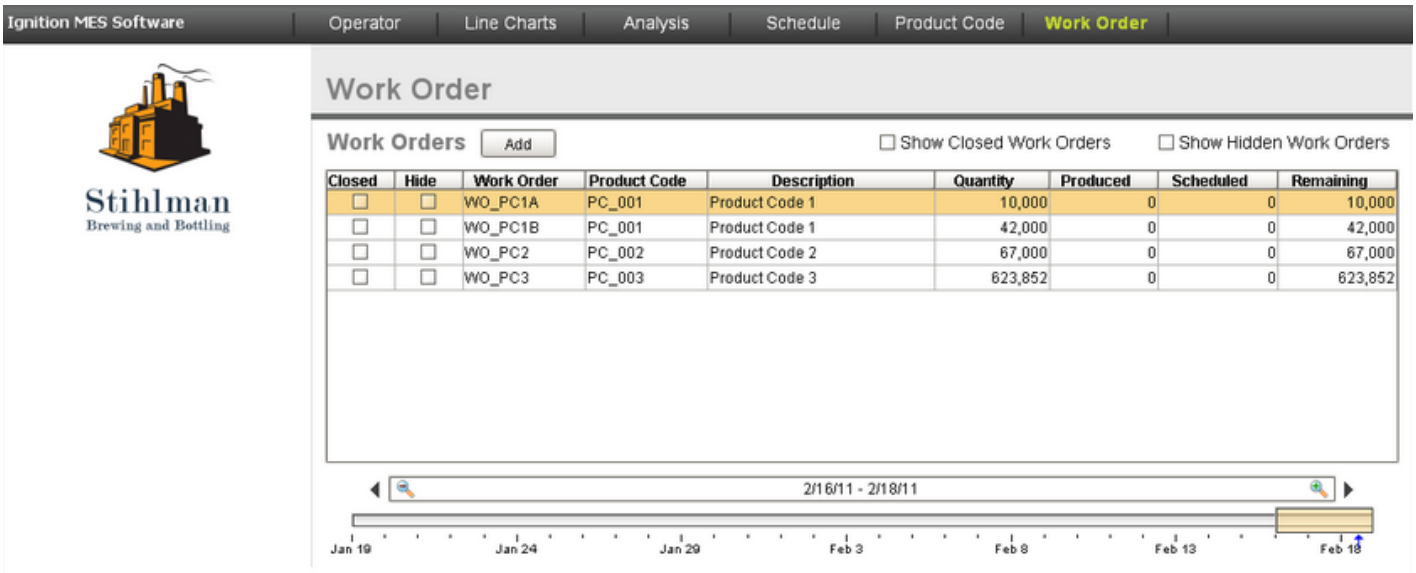
This section will walk through the user screens of the OEE Demo. As you are going through this section, it is important that you keep in mind that these screens are just provided for demo, training and to reduce the time required to get up and running. They can be modified, deleted or add new screens using the Ignition designer. But we will save that for the next section. To start the OEE Demo, go to the home section in the gateway and click on the launch button as shown below.



Launch OEE Demo Client

2.2.2.1 Work Orders

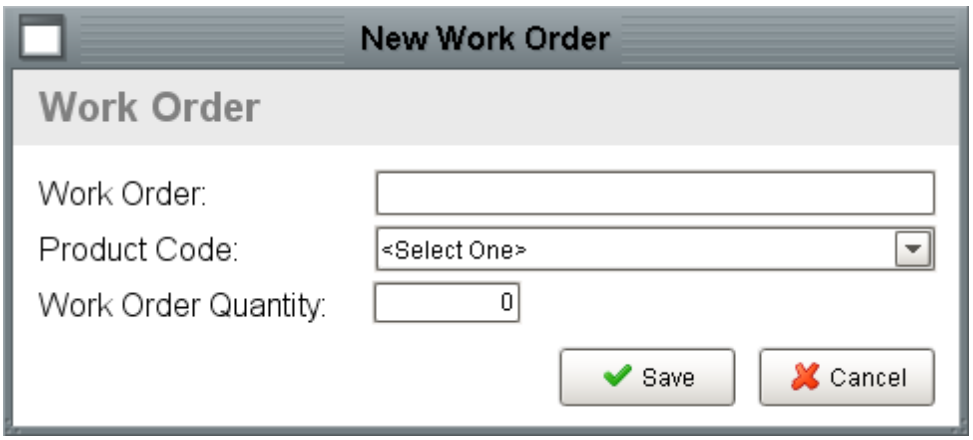
Work orders track the progress of production of a given product. They can span across multiple production runs of a given product. As shown below, the total number of products in the first work order is 10,000. We can also see that no cases have been scheduled, no cases have been produced so far, so all the cases are remaining. As production runs are scheduled and product is manufactured against a work order, these values will update.



Work Order Screen

It is possible to remove work orders that are closed or hide an open work order by selecting one of the two check boxes to the left of the work order. It is also possible to show these work orders again by selecting "Show Closed Work Orders" or "Show Hidden Work Orders" in the top, right-hand corner of the screen. In order to show work orders in a specific date or the time range, there is a slide bar at the bottom of the screen which can be dragged to the correct date. The magnifying glasses allow a more specific time or a broader range of dates to be viewed.

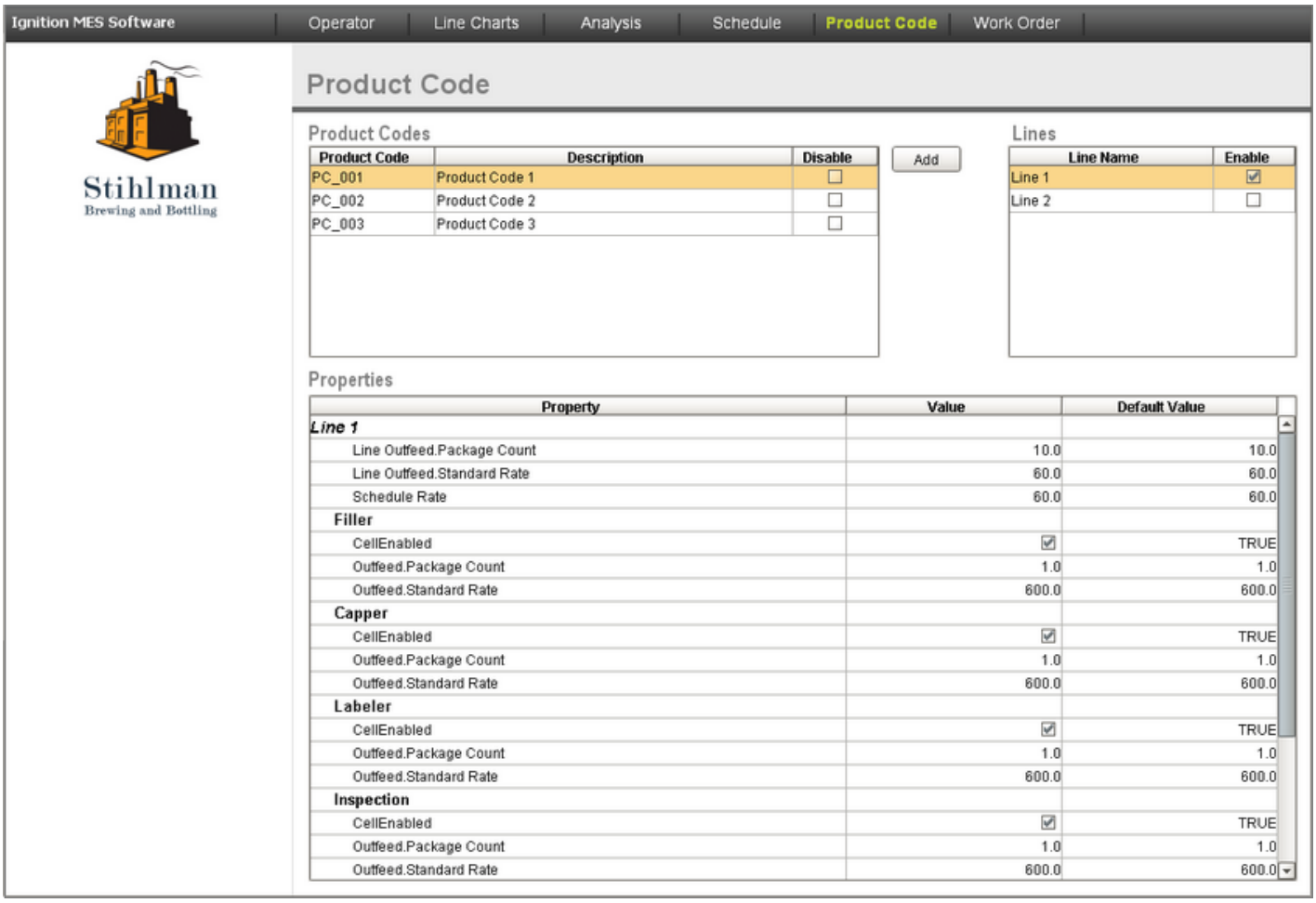
The system supports adding work orders, as well as editing and removing work orders. For companies that have ERP (Enterprise Resource Planning) or other systems containing work order information, work orders can be added or updated automatically. To add a work order, simply click "Add" and fill in the necessary information. A product code must be entered before a work order can be added.



New Work Order Window

2.2.2.2 Product Codes

Product codes or pack codes represent the products that are manufactured within your facility. If you have multiple production *lines*, this screen is where product codes are assigned to individual production *lines*. In addition, the settings for a product code may vary depending on the *line* it is being produced on. Those settings are also set on this screen in the Properties section.



Product Code Screen

To assign a product code to one or more production *lines*, select the product code, then select the "Enable" box next to the appropriate *line*. It is also possible to disable a product code altogether by selecting the box to the right of the product code. To edit the properties, select the product code and the *line* you wish to edit, then double click the value to be changed. Pressing Enter or clicking off of the value will save the change. Product code descriptions can also be edited by double clicking.

The demo demonstrates adding product codes, but the system also supports editing and removing product codes. For companies that have ERP (Enterprise Resource Planning) or other systems containing product code information, product codes can be added or updated automatically from them.

2.2.2.3 Production Schedule

The scheduling screen is similar to Outlook calendar and is easy for new users to learn how to use. It has month, week and day views that are selected by clicking on the Day **Week** Month buttons. Select the

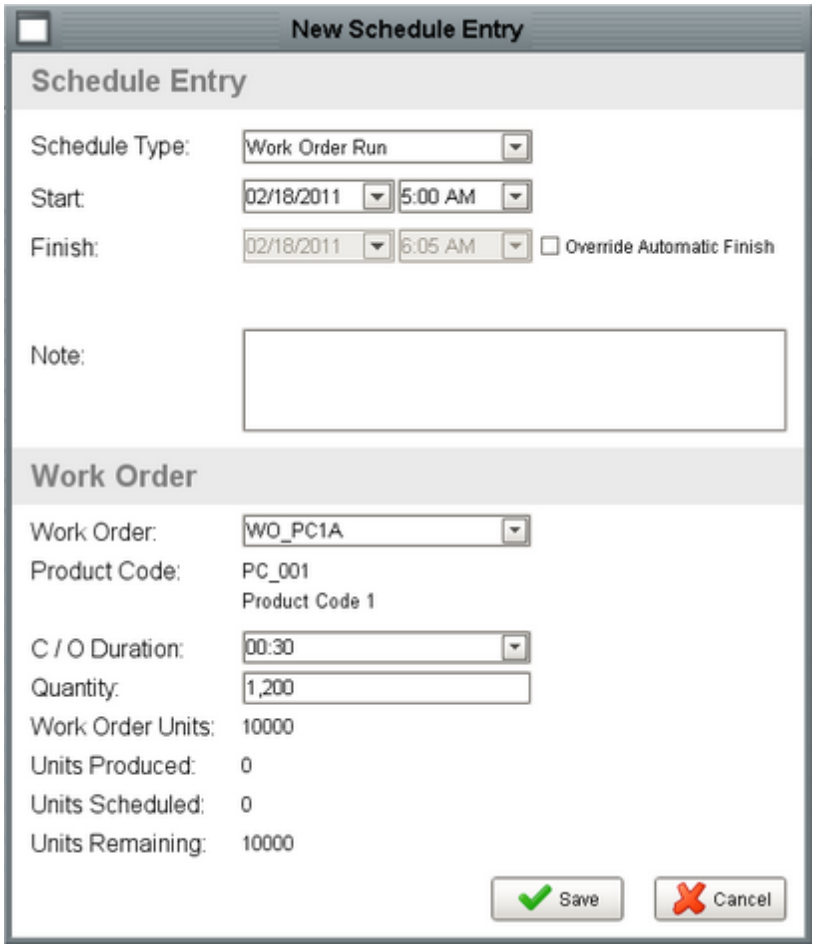
production *line* to view and edit the schedule by using the drop-down list as shown below.

Line Selection on Calendar

Production runs can be scheduled for part of a shift, across multiple shifts, days and even months.

To see how editing production run entries works, select the week view. Next, using the mouse, right-click on a day and time to start the production run and click **New Entry**. Here, you can also chose to edit or delete an existing scheduled production run. The New Schedule Entry popup window allows for scheduling production runs, maintenance and other entries. By selecting the Work Order Run schedule type, work order options appear. Continue by selecting a work order (these must be created before adding a new schedule entry). Notice, the total work order units to be produced, units produced, units scheduled and units remaining information is shown for the selected work order. The quantity entry is automatically set to the remaining units for the work order. A C/O duration may also by added to account for the change over duration before the production *line* begins.

© Inductive Automation



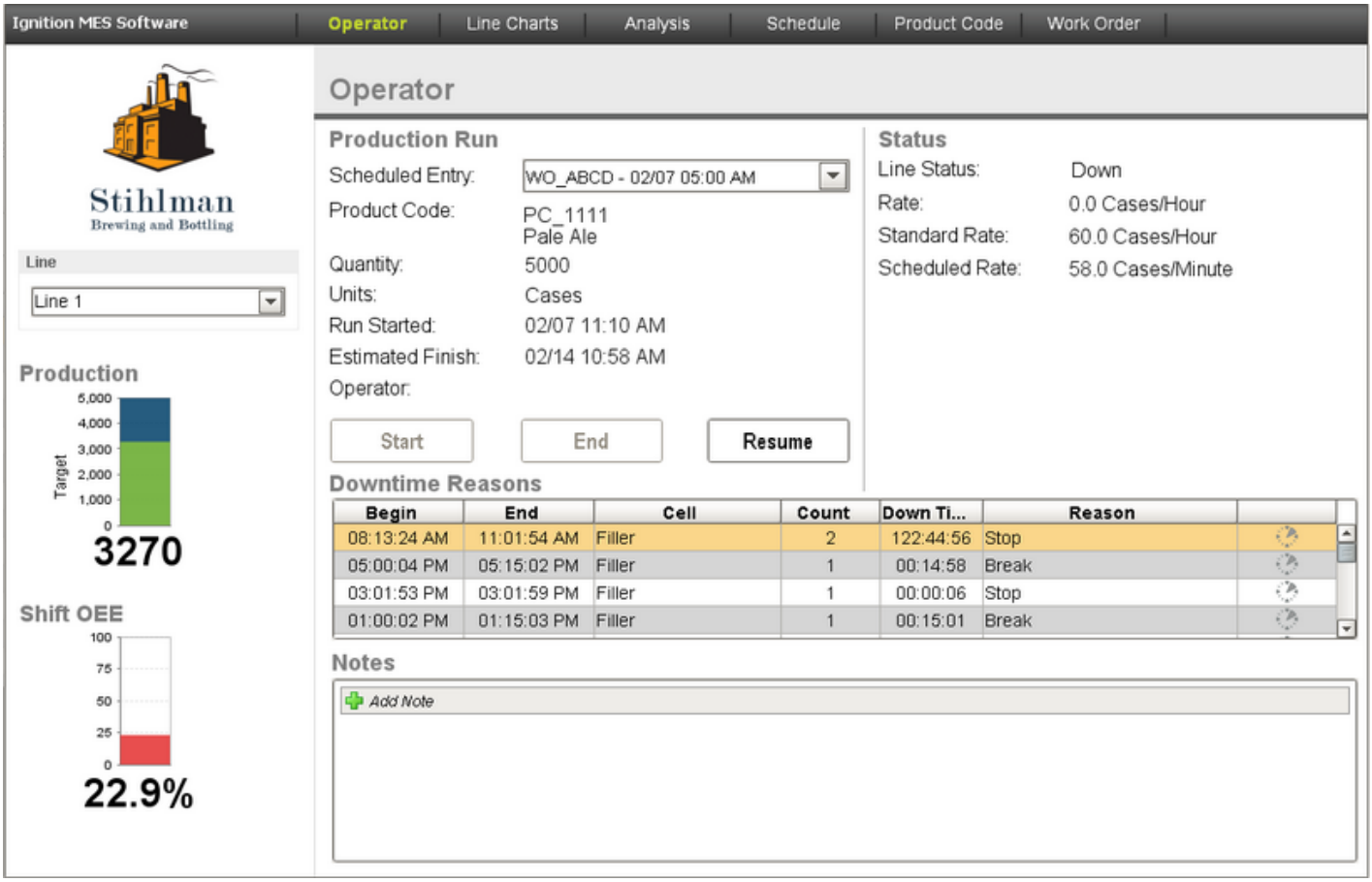
Schedule New Production Run

After entering the desired quantity, the finish time for the work order will automatically update to show the predicted production end time and date. The system forecasts the finish time based on the schedule rate for the product code associated with the work order and all breaks or meals that are configured. You can override the finish time by selecting the Override Automatic Finish option and manually selecting a date and time.

2.2.2.4 Operator Screen

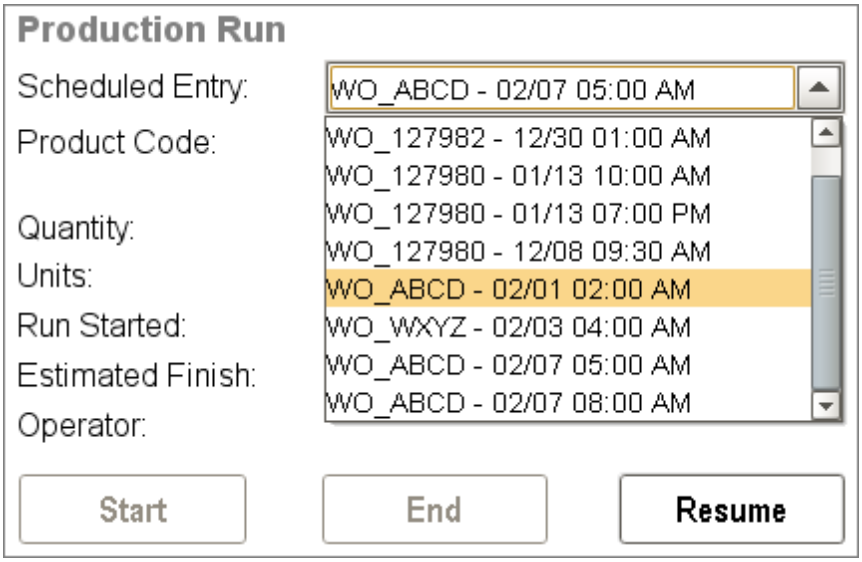
The operator screen provides an interface to allow the operator to control the current production run, select downtime reasons and monitor the current production run. This screen is used as a demo, but can reduce the amount of time needed to implement an OEE, downtime and scheduling system. It can be modified using the Ignition designer to accommodate your requirements.

© Inductive Automation



Operator Screen

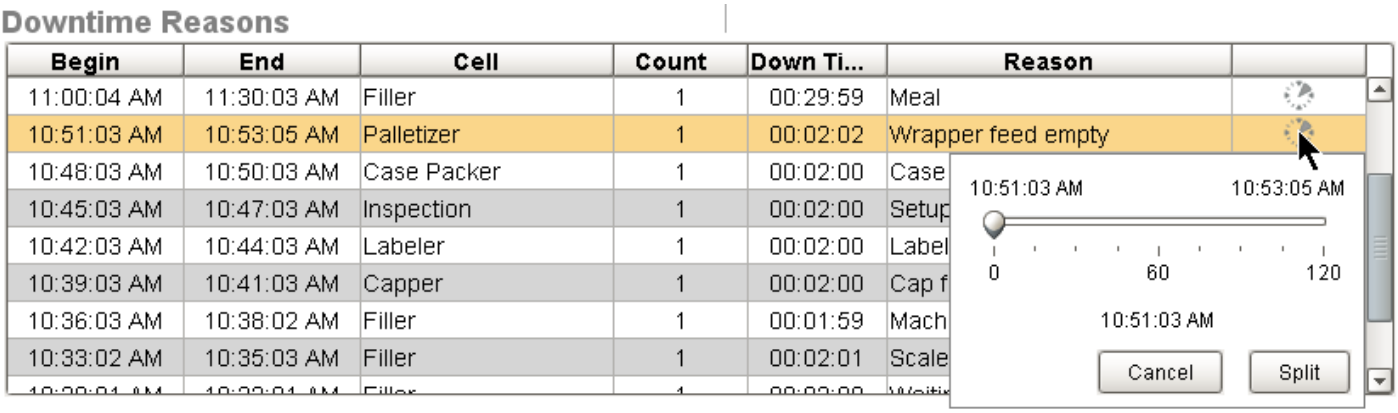
The portion of the operator screen shown below, allows the operator to start, end and resume production runs. By clicking on the down arrow of the scheduled entry drop-down box, a list of scheduled production runs that have not been started will be shown for the operator to select from. Before a different production run can be selected, the current production run must be ended.



Scheduled Run Selection

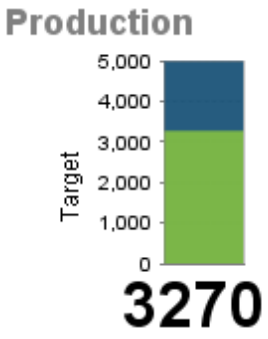
© Inductive Automation

Once the run has started, the downtime events for the selected *line* will show in the downtime reason table. Sometimes downtime events occurred for more than one reason. When this happens, downtime events can be split by clicking on the split icon as shown below. Comments can also be added by clicking to the right of the split icon.



Downtime Reason Table

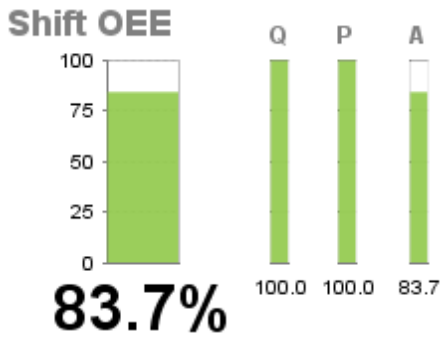
You will notice other information of the screen that give the operator realtime indication of how the production run is progressing. The indictor shown below displays the target and actual production for the current shift. At the start of every shift change, new targets are calculated for the new shift. This always provides the operator relevant production information on their shift to keep the entire production run on track.



Production Progress Indicator

Below this, the OEE information is displayed for the shift. Here, the total OEE for the shift is displayed, as well as the Quality, Performance, and Availability individually.

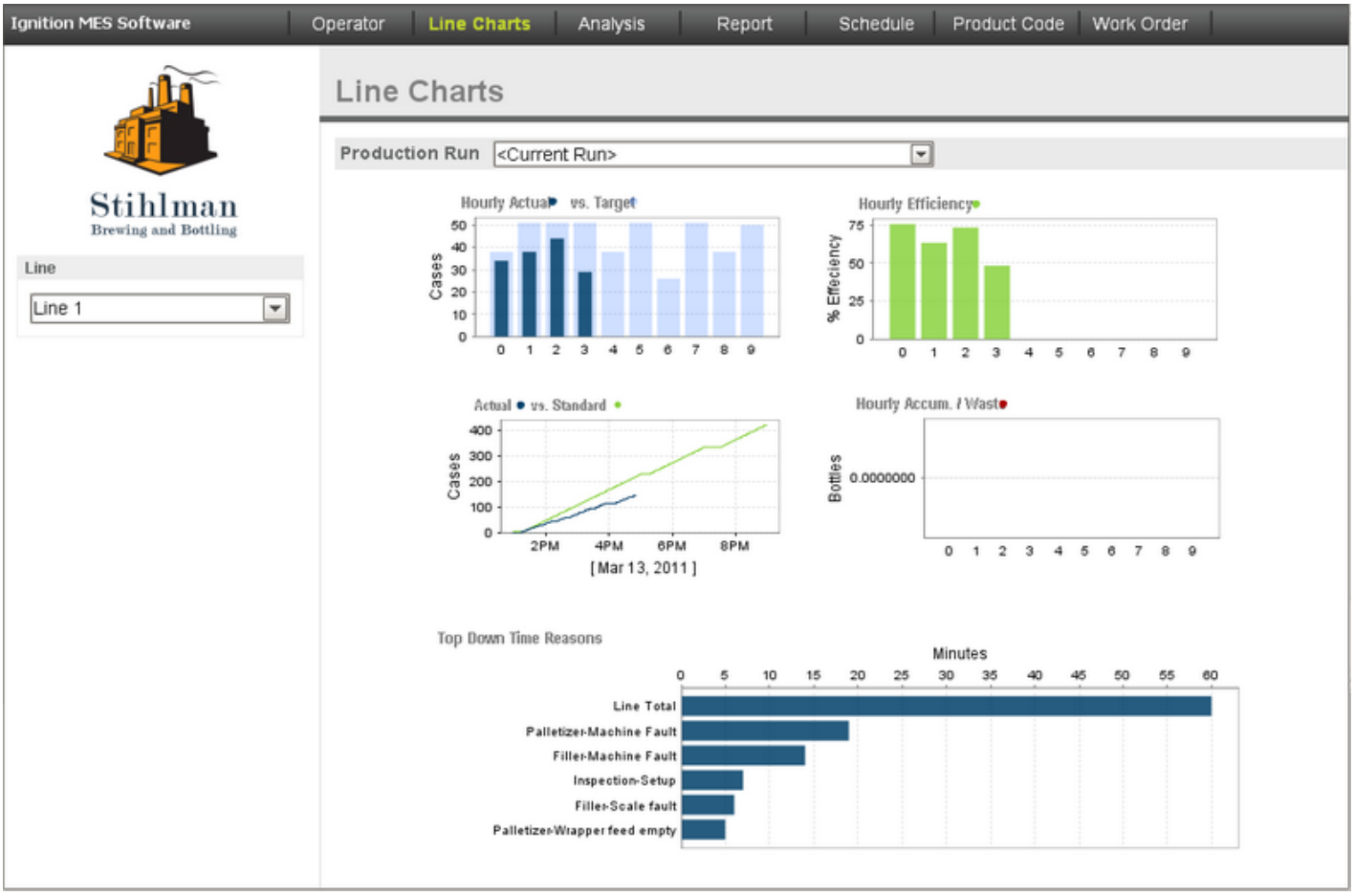
© Inductive Automation



OEE Indicator

2.2.2.5 Line Charts

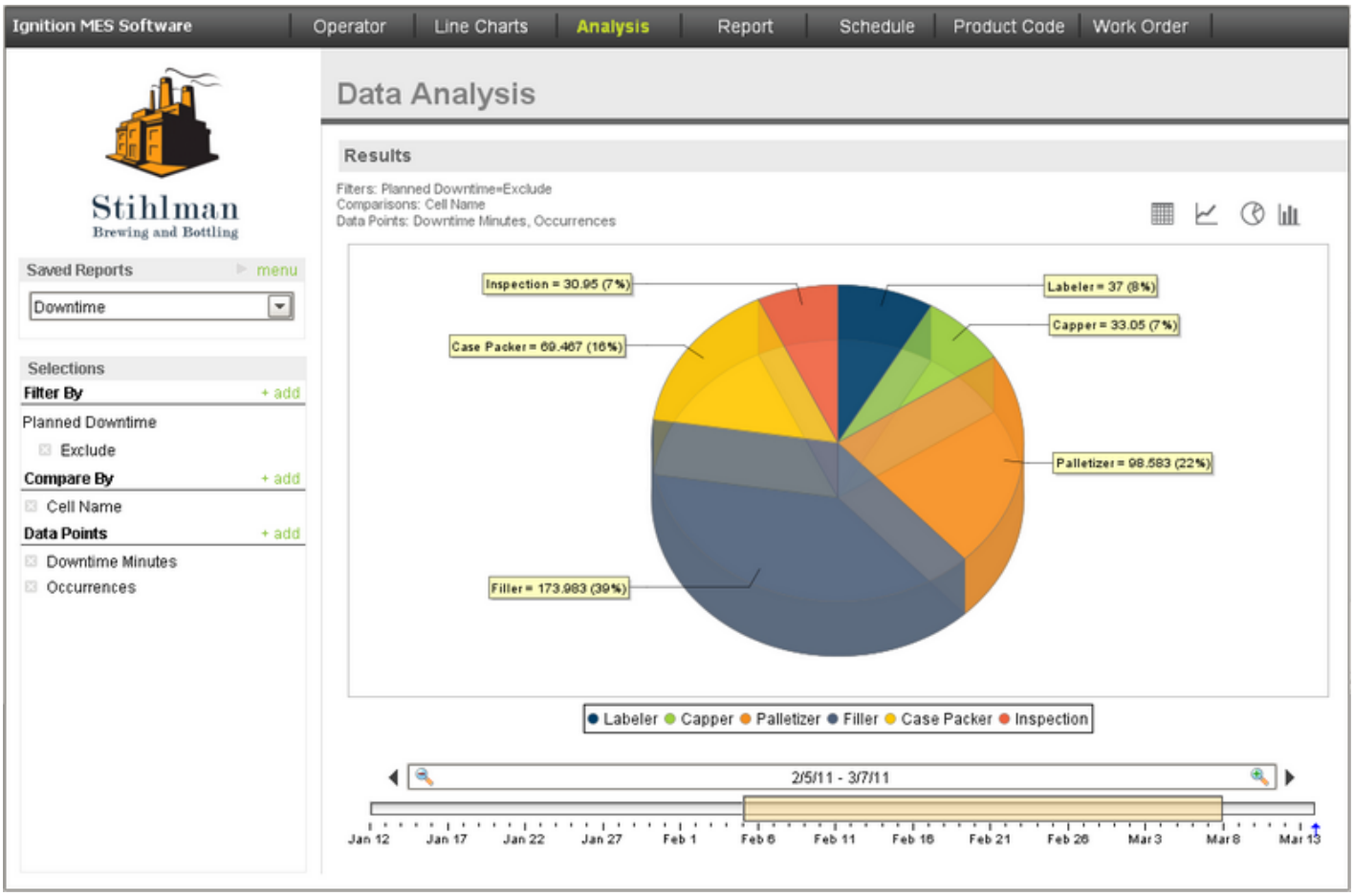
The line charts screen provides realtime information as the production runs progresses. Completed production runs can be selected using the production run drop-down list. At the bottom of the screen, a graph showing the top reasons for down time is displayed and the number of minutes of down time caused by each of the given reasons. It is possible to zoom in on any of the graphs by clicking and dragging over the desired area, or by right-clicking and selecting **Zoom In**. To restore the previous view, you can right-click and select **Zoom Out** or **Reset Axes**. **Auto Range** can also be used to see more of the graph.



Sample Line Charts Screen

2.2.2.6 Analysis Screen

The analysis screen allows for ad-hoc analysis of production data. OEE, TEEP, downtime, production and even user defined data can be viewed across a date range. This data can also be filtered to only include specific criteria. Additionally, comparisons can be made between different factors. For example, downtime by operator can be analyzed or even downtime by operator by shift.

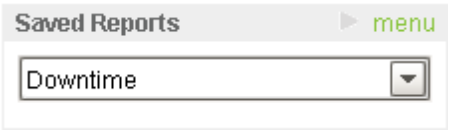


Ad-hoc Analysis Screen

The date range selector at the bottom is used to define the data range to include in the analysis. As you change the start or end dates, only the production runs that are within that range will be included in the analysis.

Stored Analysis

In the demo project, there are pre-configured analyses that can be selected in the store reports section. As different stored analyses are selected, the values in the Selections section will change. The demo project, has a Downtime by Line stored analysis. Selecting it will make the selection to view downtime occurrences and downtime minutes values by individual *lines*.



Saved Reports

© Inductive Automation

If the pie chart is selected, only the downtime occurrences will be represented graphically. However, if the bar chart is selected, then both the downtime occurrences and the downtime length in minutes will be shown graphically in separate bars.

Filter By

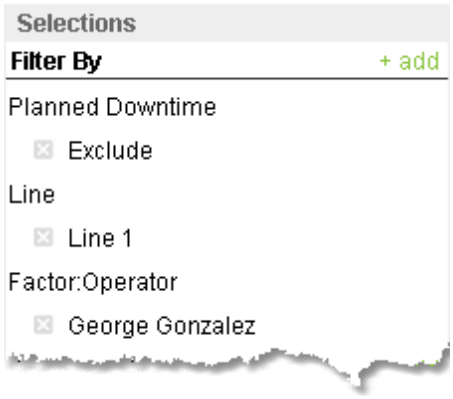
Once an stored analysis has been selected, you can change the selections to zero in on the data that you desire. The filter section allows you to limit the data that is included in the analysis. Filters can be added by clicking on the + add icon on the right side of the *Filter By* section. Within the popup filter selection window, scroll down to the **Factor:Operator** option and click the icon. Notice the names of operator that can be selected. Clicking on George Gonzalez will add the Factor:Operator = George Gonzalez causing the analysis results to included downtime data while George Gonzalez was the operator.

Filter By Options

The list of available filters change based on the date range. For example, if George Gonzalez was on vacation for the desired date range, then his name will not show as an available options.

The filter selection shown below includes data for only Line 1 when George Gonzalez was the operator and excluding all planned downtime.

© Inductive Automation



Filter By Selections

Filter By items can be removed by clicking on the ⊠ located to the left of the name.

Compare By

Breaking up information into groups is more meaningful than just seeing a total for a given date range. For example, knowing the total downtime for Line 1 for a given data range really does not provide actionable information that can be used to improve efficiencies. Now, by comparing the total downtime for each machine on Line 1, it is possible to identify the machine(s) causing the most downtime. Focusing efforts on these machines and solving sources of downtime will result in better efficiencies.

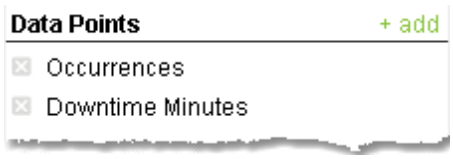
Additional Compare By items can be added by clicking on the + add icon on the right side of the *Compare By* section. Within the popup Compare By selection window, click on the desire item that you want to compare analysis results between.

Compare By Selections

Compare By items can be removed by clicking on the ⊠ located to the left of the name.

Data Points

Data points are the individual pieces of information that will be present in the analysis. For example, downtime minutes or downtime occurrences are just two of the many available data points. To add a data point, click on the + add icon on the right side of the *Data Points* section. Within the popup Data Point selection window, click on the data point item to include in the analysis.



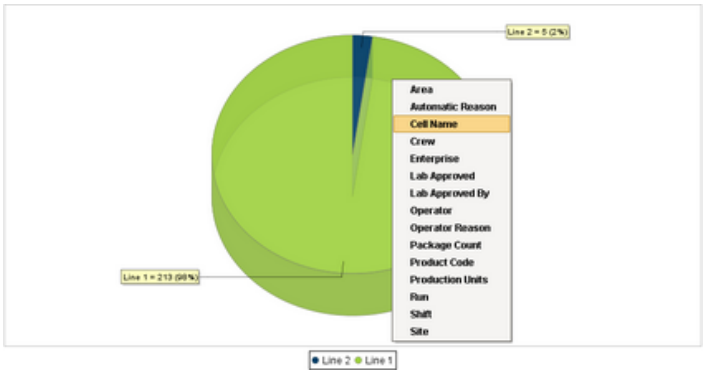
Data Point Selections

Data Points can be removed by clicking on the ⊠ located to the left of the name.

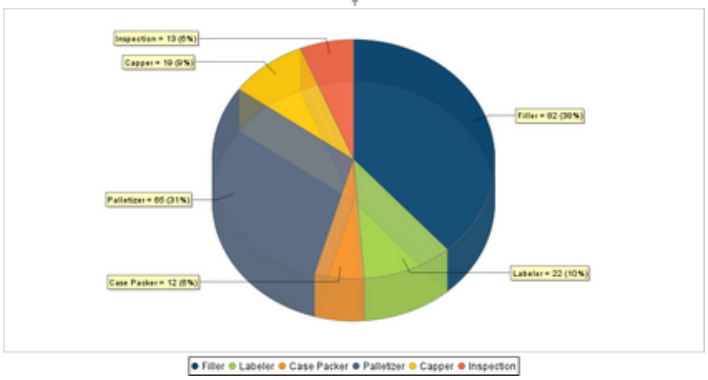
© Inductive Automation

Drill Down

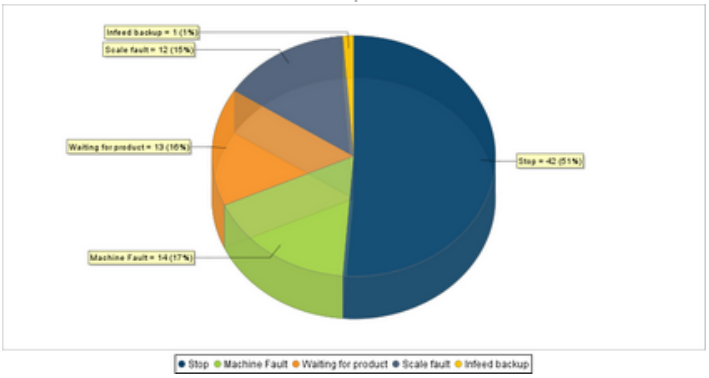
The drill down feature simplifies the compare by and filter selections. Click on a chart series to display the available drill down options. As shown in **Drill Down Example 1** below, clicking on the Line 1 pie segment will show a popup menu of drill down options. If the **Cell Name** option is selected, then the analysis filters to show the information by Cell Name . The Filter By and the Compare By sections add *Cell Name*. The result is shown in **Drill Down Example 2**. Again, by clicking on the "Filler" pie segment and selecting **Operator Reason**, the Filter By and Compare By selections will change to show information for only Line 1 Filler and Compare By Operator Reason as shown in **Drill Down Example 3**.



Drill Down Example 1



Drill Down Example 2



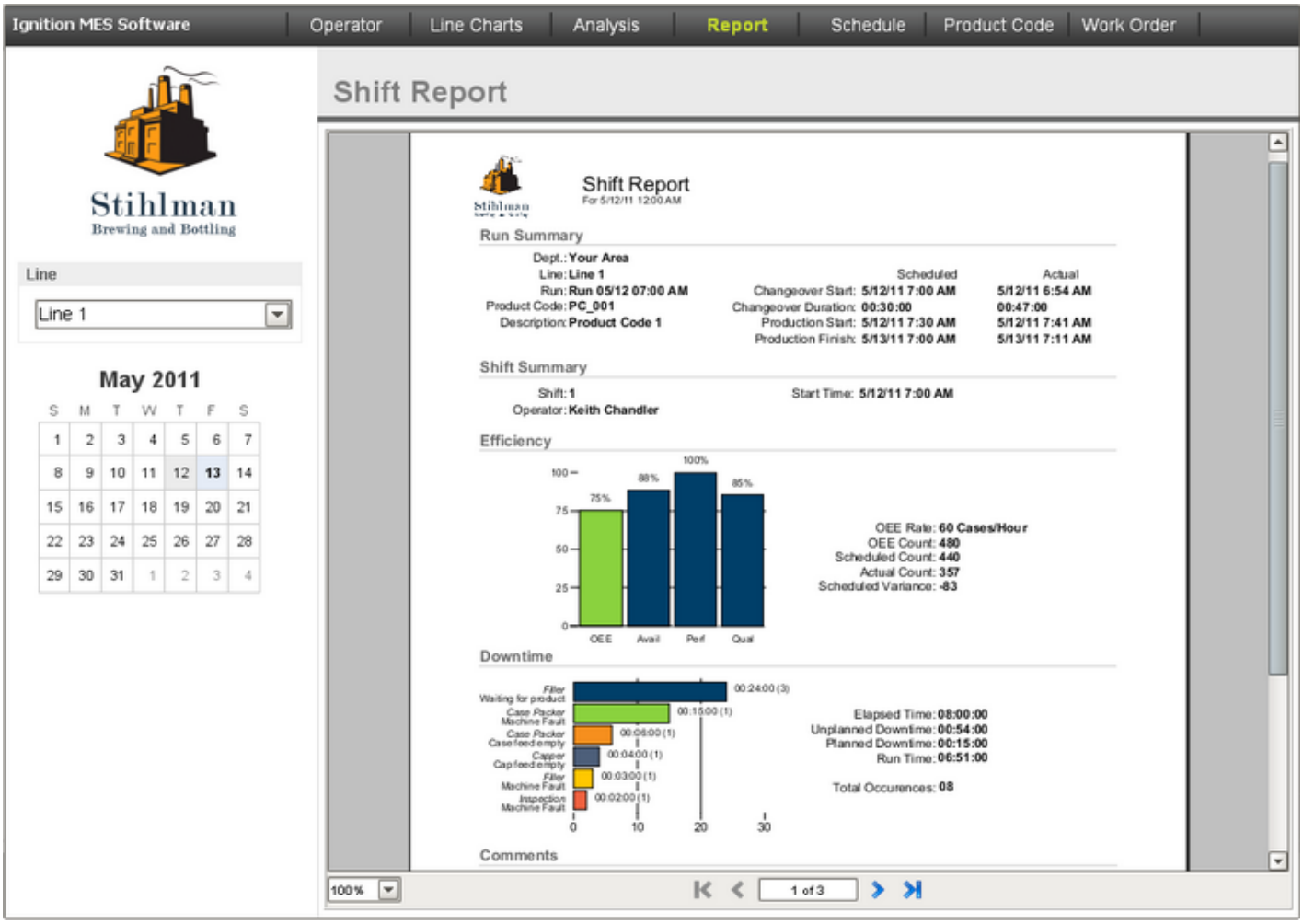
Drill Down Example 3

© Inductive Automation

2.2.2.7 Report Screen

This is a very basic sample report that shows downtime information for a *line* during a given date range. It can be expanded to include much more information. It is built using the Ignition Reporting Module and presents data provided by the OEE Downtime and Scheduling module. All the flexibility of how data is presented in the analysis screen is also available in reports and multiple analysis results can be included in reports.

In addition to viewing reports in a screen, they can be printed, saved to PDF, HTML or image.



Sample Report

2.2.3 Dashboard

The executive dashboard provides a high level view of OEE, downtime, and production information.

2.2.3.1 Area Summary

The **Dashboard Area Summary** shows OEE, Downtime, and Production and Waste Count comparisons between different *areas* through various graphs. The slide at the bottom of the page causes the date to be changed, allowing the user to select a specific day or time, or a broad range of dates to view. These graphs may also be saved or printed by right-clicking on the desired graph.

© Inductive Automation

Area Summary Screen

Line Graph Settings

There are many settings for lines graphs which can be adjusted by right-clicking on a line graph. These options can be used to improve graphs and select specific areas to save and print.

Mode

Zoom: Allows the user to click and drag over the specific area of the graph that he or she wants to enlarge.

Pan: Allows the user to click and drag to move to a different area of the graph.

Mark: Gives the domain and range values of a single line at 12:00 AM on the day selected by clicking on the graph.

X-trace: Gives the domain value of all lines on the graph at 12:00 AM on the day selected by clicking on the graph.

Background

The background color of a line graph is white by default, but may be changed to black to allow the user to see lighter colors.

Zoom

Allows the user to zoom in or out along the domain axis, the range axis, or both axes. Can also be used with bar graphs.

Auto Range

Automatically selects the best range of values for the domain axis, the range axis, or both axes to in order to see the entire graph for the date or time range selected at the bottom of the screen. Can also be used with bar graphs.

Reset Axes

Resets both axes to their original ranges.

© Inductive Automation

2.2.3.2 Enterprise Summary

Similar to the Area Summary, the **Dashboard Enterprise Summary** compares OEE, Production Counts, Downtime, and Waste Counts between different *enterprises* in a *production*, or a single *enterprise* over time. The slide at the bottom of the page causes the date to be changed, allowing the user to select a specific day or time, or a broad range of dates to view. These graphs may also be saved or printed by right-clicking on the desired graph.

Enterprise Summary Screen

Line Graph Settings

There are many settings for lines graphs which can be adjusted by right-clicking on a line graph. These options can be used to improve graphs and select specific areas to save and print.

Mode

Zoom: Allows the user to click and drag over the specific area of the graph that he or she wants to enlarge.

Pan: Allows the user to click and drag to move to a different area of the graph.

Mark: Gives the domain and range values of a single line at 12:00 AM on the day selected by clicking on the graph.

X-trace: Gives the domain value of all lines on the graph at 12:00 AM on the day selected by clicking on the graph.

Background

The background color of a line graph is white by default, but may be changed to black to allow the user to see lighter colors.

Zoom

Allows the user to zoom in or out along the domain axis, the range axis, or both axes. Can also be used with bar graphs.

© Inductive Automation

Auto Range

Automatically selects the best range of values for the domain axis, the range axis, or both axes to in order to see the entire graph for the date or time range selected at the bottom of the screen. Can also be used with bar graphs.

Reset Axes

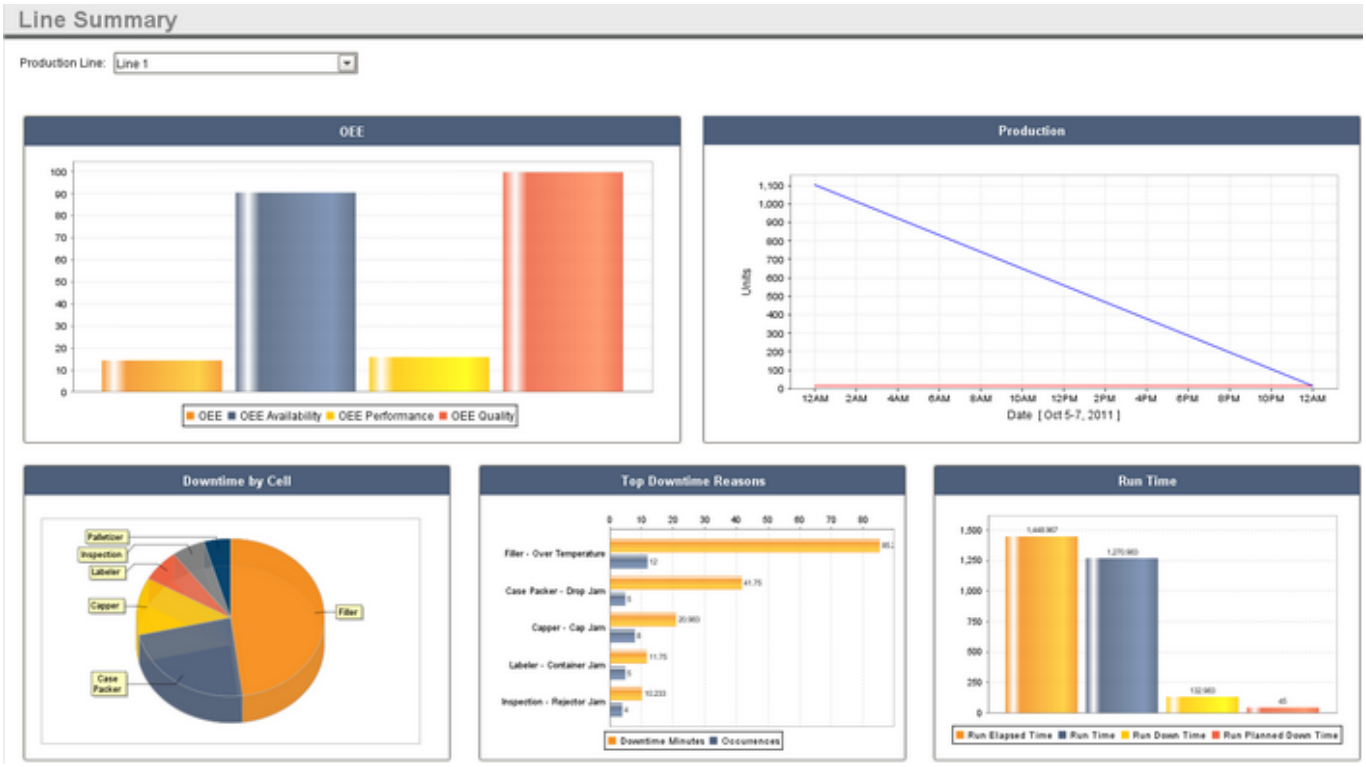
Resets both axes to their original ranges.

2.2.3.3 Impromptu Analysis

See Data Analysis.

2.2.3.4 Line Summary

The **Dashboard Line Summary** is also similar to the Dashboard Area Summary. After selecting a *line*, the OEE, Production Counts, and Run Time for that *line* can be viewed. Downtime by Cell is also available on this screen, showing which *cells* could by upgraded or changed to improve efficiency. Another important element on this screen is the Top Downtime Reasons graph, which displays downtime reasons, such as a labeler running out of labels, and the total amount of downtime that is caused each event. This increases the amount of information that can used to increase efficiency. The slide at the bottom of the page causes the date to be changed, allowing the user to select a specific day or time, or a broad range of dates to view. These graphs may also be saved or printed by right-clicking on the desired graph.



Line Summary Screen

Line Graph Settings

There are many settings for lines graphs which can be adjusted by right-clicking on a line graph. These options can be used to improve graphs and select specific areas to save and print.

Mode

Zoom: Allows the user to click and drag over the specific area of the graph that he or she wants to enlarge.

Pan: Allows the user to click and drag to move to a different area of the graph.

Mark: Gives the domain and range values of a single line at 12:00 AM on the day selected by clicking on the graph.

X-trace: Gives the domain value of all lines on the graph at 12:00 AM on the day selected by clicking on the graph.

Background

The background color of a line graph is white by default, but may be changed to black to allow the user to see lighter colors.

Zoom

Allows the user to zoom in or out along the domain axis, the range axis, or both axes. Can also be used with bar graphs.

Auto Range

Automatically selects the best range of values for the domain axis, the range axis, or both axes to in order to see the entire graph for the date or time range selected at the bottom of the screen. Can also be used with bar graphs.

Reset Axes

Resets both axes to their original ranges.

2.2.3.5 Site Summary

The **Dashboard Site Summary** shows OEE, Production Counts, Downtime, and Waste Count comparisons between different *sites* through bar or line graphs. The slide at the bottom of the page causes the date to be changed, allowing the user to select a specific day or time, or a broad range of dates to view. These graphs may also be saved or printed by right-clicking on the desired graph.

Site Summary Screen

Line Graph Settings

There are many settings for lines graphs which can be adjusted by right-clicking on a line graph. These options can be used to improve graphs and select specific areas to save and print.

Mode

Zoom: Allows the user to click and drag over the specific area of the graph that he or she wants to enlarge.

Pan: Allows the user to click and drag to move to a different area of the graph.

Mark: Gives the domain and range values of a single line at 12:00 AM on the day selected by clicking on the graph.

X-trace: Gives the domain value of all lines on the graph at 12:00 AM on the day selected by clicking on the graph.

Background

The background color of a line graph is white by default, but may be changed to black to allow the user to see lighter colors.

Zoom

Allows the user to zoom in or out along the domain axis, the range axis, or both axes. Can also be used with bar graphs.

Auto Range

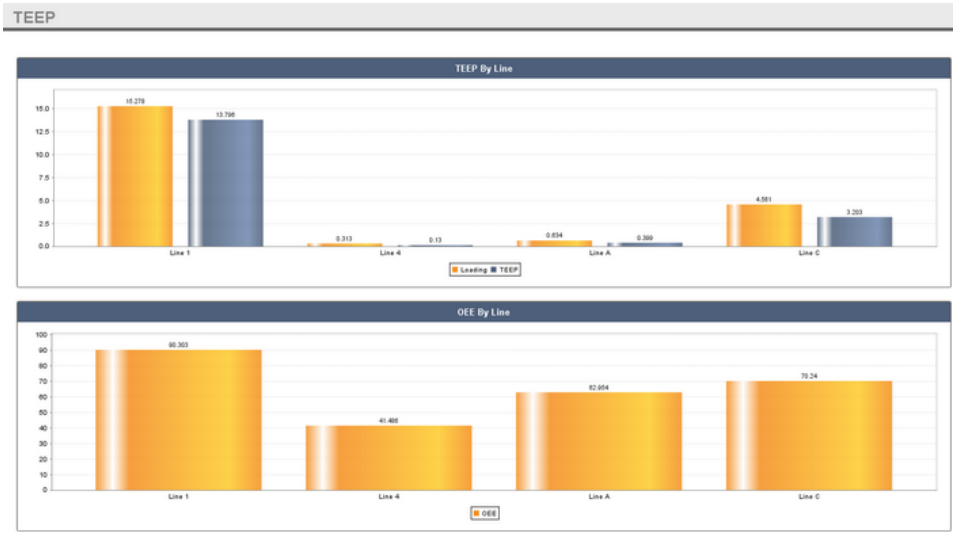
Automatically selects the best range of values for the domain axis, the range axis, or both axes to in order to see the entire graph for the date or time range selected at the bottom of the screen. Can also be used with bar graphs.

Reset Axes

Resets both axes to their original ranges.

2.2.3.6 TEEP

The Dashboard TEEP screen compares the TEEP and the OEE between all lines with production data though bar graphs. The slide at the bottom of the page causes the date to be changed, allowing the user to select a specific day or time, or a broad range of dates to view. These graphs may also be saved or printed by right-clicking on the desired graph. For more information on TEEP, click here. For more information on OEE, click here.



TEEP Screen

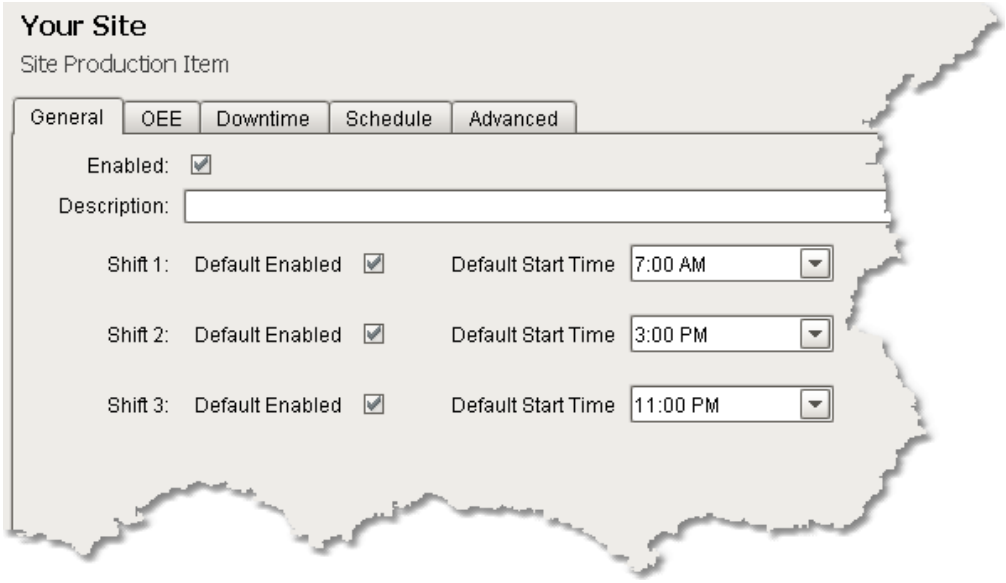
2.2.4 Production Model

A production model defines your manufacturing or process in tree view form. It enables an organized manor to easily configure, control and analysis your facility. See Production Model for more detailed information.

2.2.4.1 Production Item Settings

General Settings

When you click on the "Your Site" production item in the production model, there are settings that are accessible in the open workspace. By clicking the ***General*** tab, the current general settings are visible and can be changed. As shown below the *Default Shift Start Time* settings are visible and can be changed.



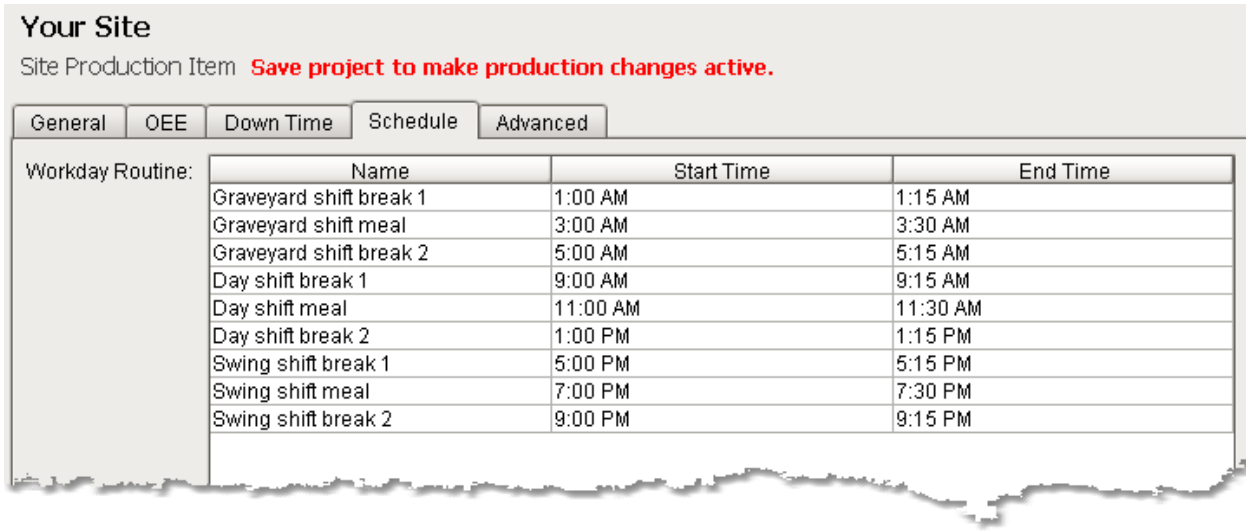
Default Shift Start Time Settings

When configuring a production *Area* the Shift Start Times can be inherited from the production *Site* or overridden. The same is true for production *Lines*.

Workday Routine Settings

From the **Schedule** tab, daily activities that are considered *scheduled downtime* can be entered. This includes activities such as breaks, meals, safety meetings, etc. When production runs are scheduled they are scheduled around these activities.

© Inductive Automation

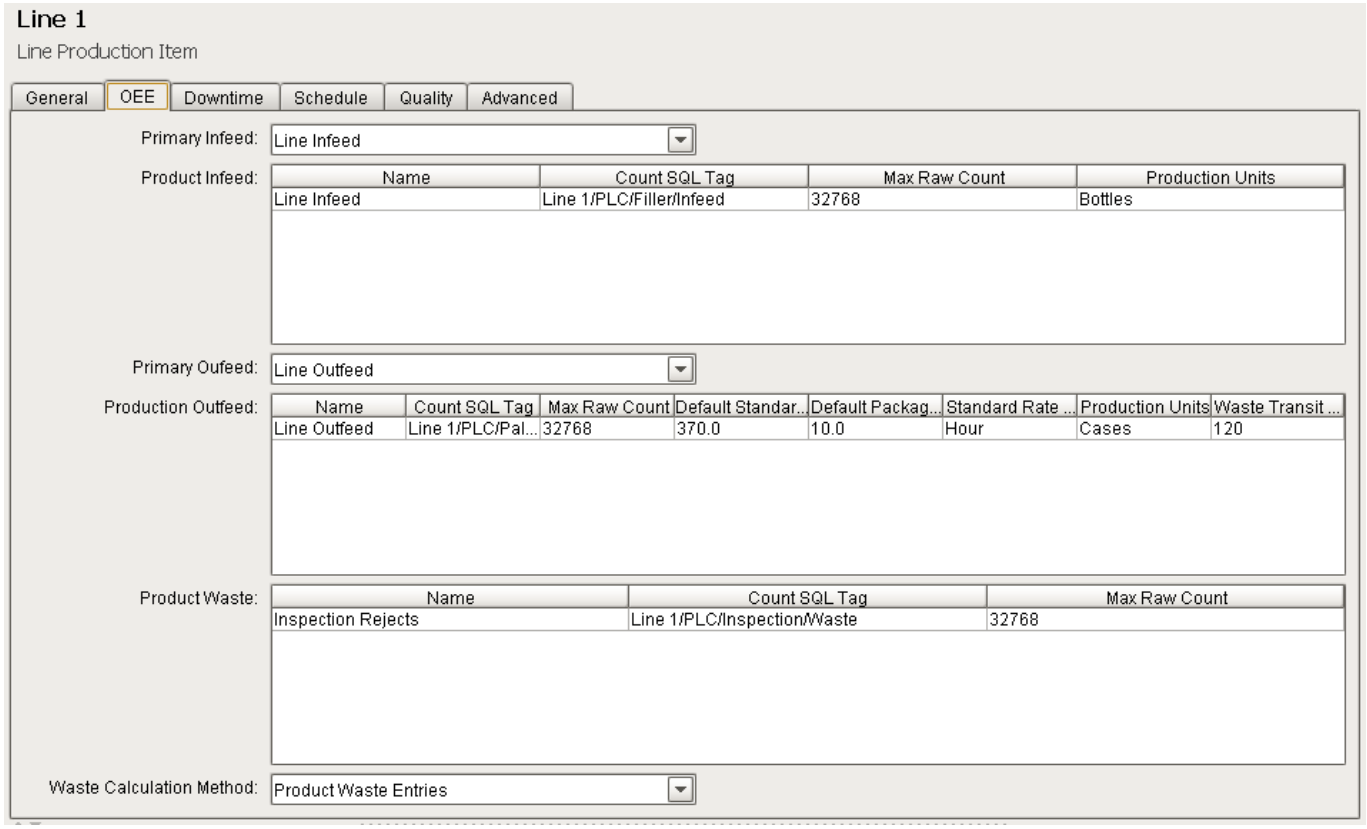


Workday Routine List

These settings can be inherited or overridden by a production *Area*. A production *Line* can in turn inherit or override the entries from the production *Area*.

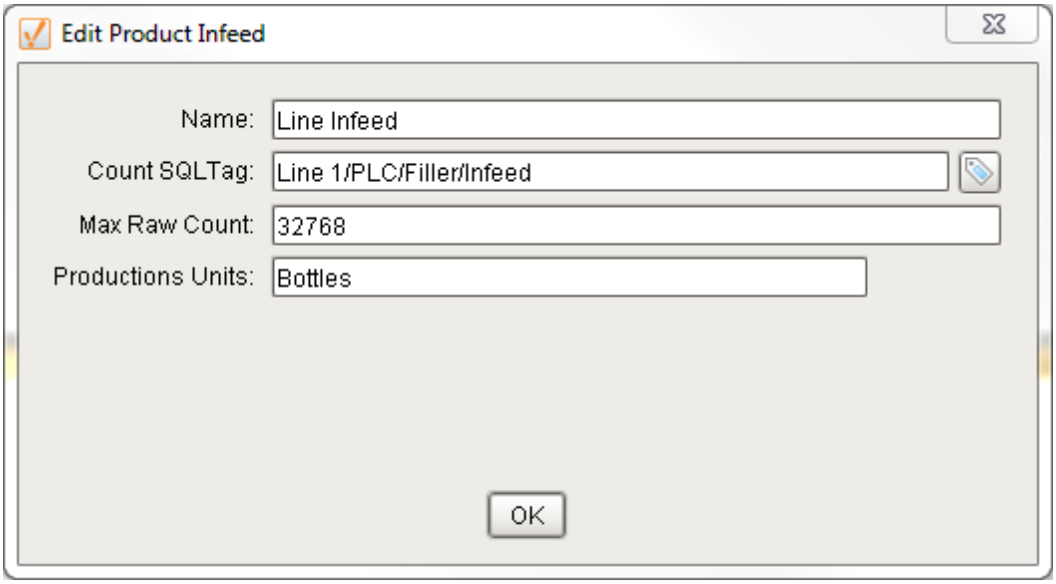
OEE Settings

For OEE values to be calculated, production data is needed. This is configured by clicking on the ***OEE*** tab and configuring the system to collect production counts. True OEE calculations use product infeed counts to determine OEE Performance. The production outfeed is used to determine waste which affects OEE Quality. At a minimum, a product outfeed for the production *line* must be configured, but for more accurate results, a product infeed should be configured as well.



Line OEE Settings

Optionally, each cell can be configured with product outfeed and infeeds. This will enable OEE data to be calculated for each cell as well as the production *line*.



Product Infeed Settings

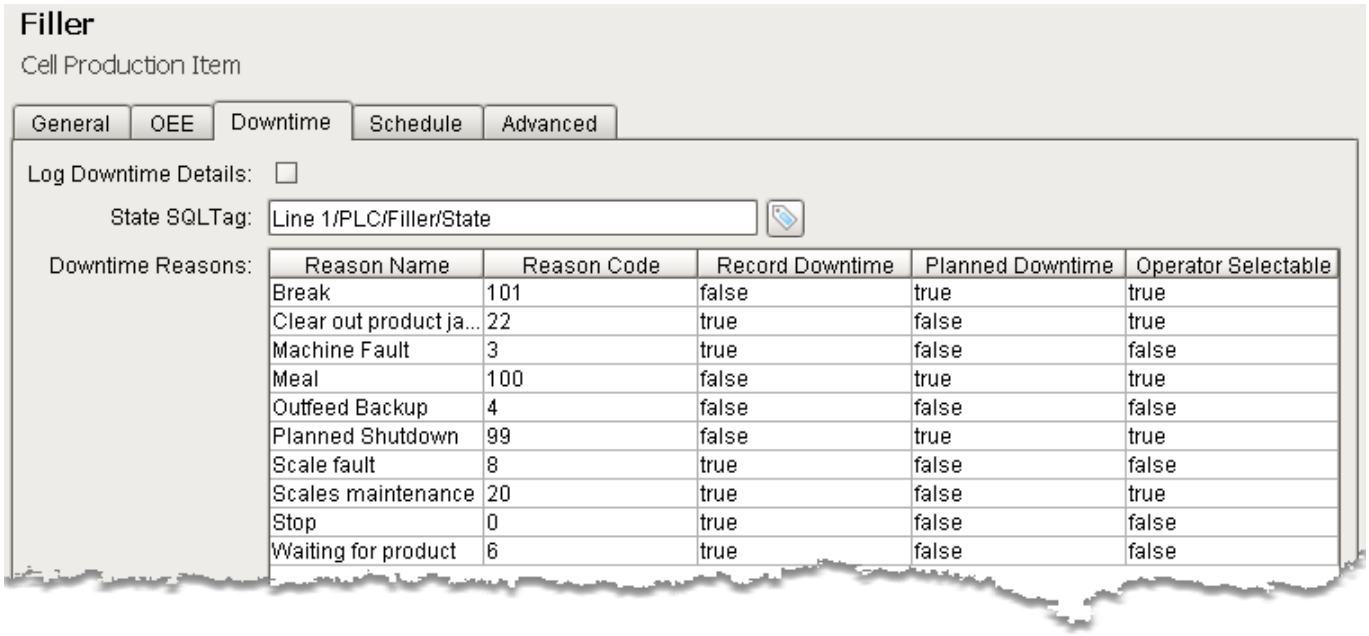
The Count SQL Tag property is set to the Ignition SQLTag that will provide infeed production counts. This is typically from a PLC, but can be from a barcode scanner, database or other source. The programming required in the PLC is simplified greatly because no handshaking or start of resets are required. In the PLC, the counter can simply rollover from the maximum value of a counter back to 0 and continue counting. The OEE module tracks the production count at the start of the run and all rollovers. This tracking is even maintained during power outages.

Downtime Settings

The OEE Downtime and Scheduling module uses a single numeric value, typically read from a PLC, to determine the current state. This applies to both a production *line* or production cells of a *line*. If the state value is 0, it is considered that the *line* or cell is idle and if it is 1, it is considered the *line* or cell is running. State values 2 on up (typically to 32767), are user defined and can be automatically detected or can tied to a operator selectable downtime reason.

Below is a list containing sample downtime reasons. Notice the ***Record Downtime***, ***Planned Downtime*** and ***Operator Selectable*** columns. If the ***Record Downtime*** option is true, then downtime events with this reason will be treated as unplanned downtime. This allows for downtime reasons such as *outfeed backup* to not be counted as unplanned downtime. If the ***Planned Downtime*** option is true, then downtime events with this reason will be treated as planned downtime.

© Inductive Automation



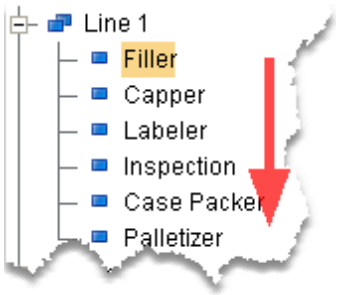
Reason Name	Reason Code	Record Downtime	Planned Downtime	Operator Selectable
Break	101	false	true	true
Clear out product ja...	22	true	false	true
Machine Fault	3	true	false	false
Meal	100	false	true	true
Outfeed Backup	4	false	false	false
Planned Shutdown	99	false	true	true
Scale fault	8	true	false	false
Scales maintenance	20	true	false	true
Stop	0	true	false	false
Waiting for product	6	true	false	false

Cell Downtime List

Downtime tracking can be done three different methods. The first two methods focus on the primary reason the production *line* is not producing product. The third method tracks all downtime for production cell regardless of whether it caused production loss for the *line*.

In cases where there is a single PLC controlling the production *line*, downtime events can be read from a single numeric value representing the *line* state. The State SQLTag and downtime reasons are configured in the production *line*.

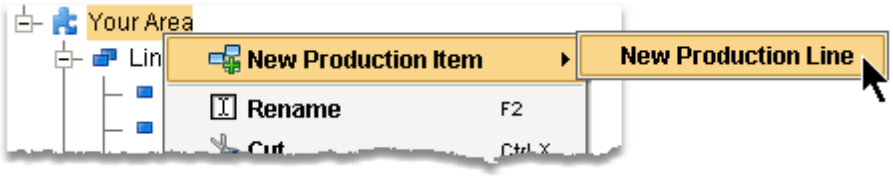
It is common that each cell of a production *line* has its own PLC. To set up communication between the PLC and a master PLC, and to add the logic to determine the cause of why production *line* is not producing product is a complex process; however, the OEE Downtime module eliminates this complexity with a feature called **Use Key Reason Detection.** When this option is selected on the **Downtime** tab for a *line*, the module will determine the primary cause as to why the *line* is not producing product. This method uses the flow of the *line* to determine the cause for the *line* not being able to product. It also assumes there is a primary cell that, if down, will cause the *line* to stop producing product. If the first cell is down for a reason that is not configured as *Record Downtime*, the next cell will be checked. If it is down for a reason that is configured as *Record Downtime*, then it will be assigned as *line* downtime cell and reason. When the second cell that caused the *line* downtime restarts but the first cell has not started yet because its discharge is still backed up, then the original cell and reason will still be the cause until the first cell restarts.



Flow of Key Reason Detection

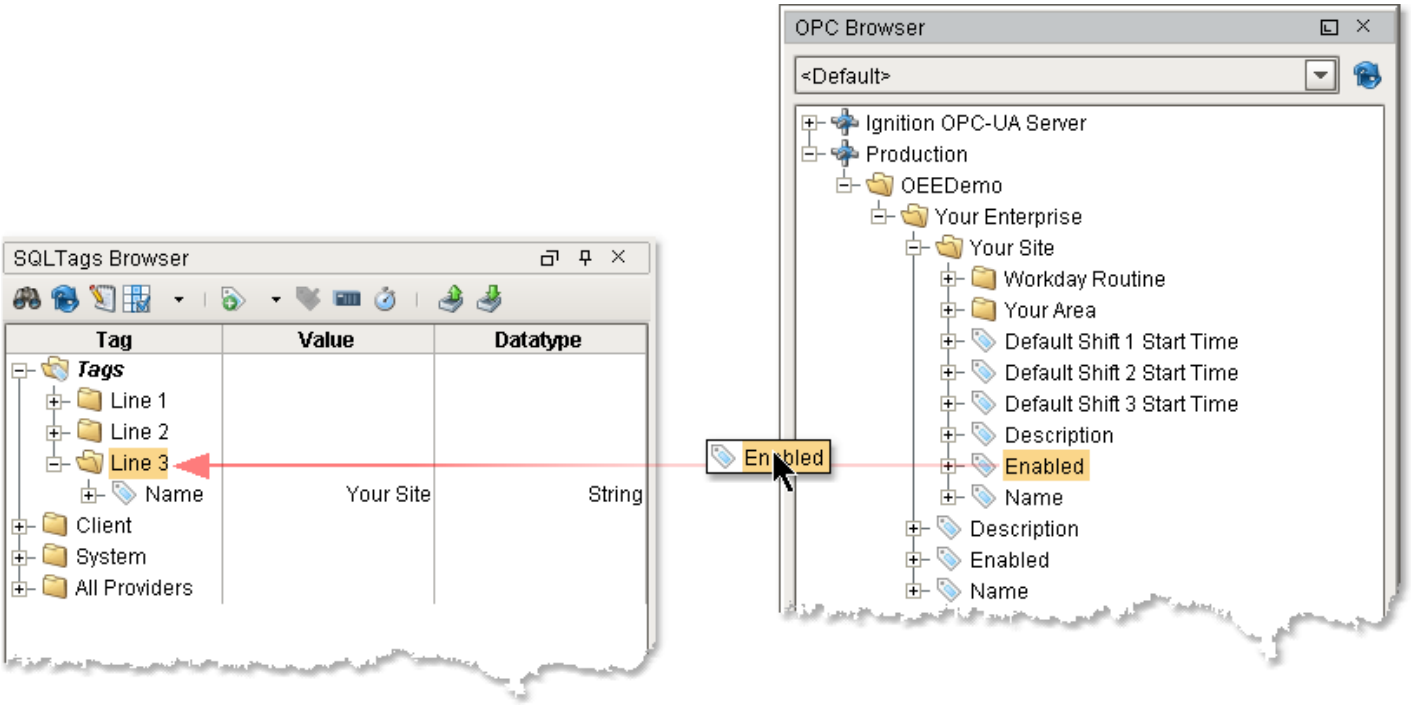
2.2.4.2 Adding Production Items

New production model items can be added by right clicking on the parent item. A popup menu with the available options will appear. For example, right clicking on the "Your Area" production item, then clicking on the **New Production Item > New Production Line** menu item will add a new *line* below "Your Area."



Adding A New Production Line

After production items are added, their OEE downtime and scheduling configuration settings and runtime values are available for use in Ignition windows, transaction groups, scripting, etc. Before values from the Production OPC Server can be used, they must be added to the Ignition SQLTags. This is done in the designer by selecting the SQLTags Browser and clicking on the icon. This will cause the OPC Browser to appear. Next, drill down in the **Production** node within the OPC Browser. Drag any of the Production OPC Values over to the SQLTags Browser as depicted below.



Add Production OPC Server Values to SQLTags

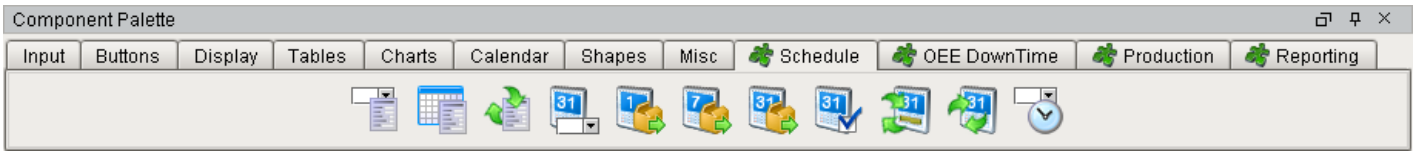
2.2.5 Configuration

Because the OEE Downtime and Scheduling Module is built on the Ignition platform, configuration is done using the Ignition Designer.

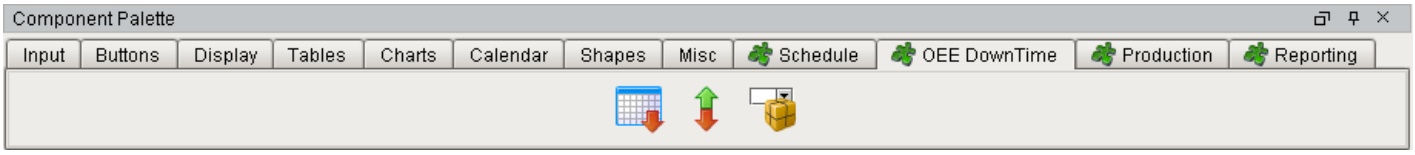
© Inductive Automation

2.2.5.1 Components

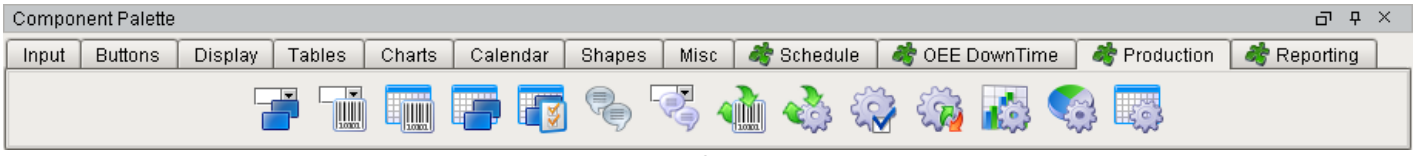
In addition to the components that come with Ignition, the OEE Downtime and Scheduling Module provides additional components that make implementing an OEE, downtime and scheduling system easier. These components greatly reduce, or in some cases eliminate, the need for custom SQL statements and scripting.



Schedule Components

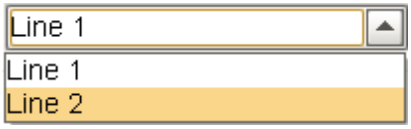


OEE and Downtime Components



Production Components

For example the Production Line Selector component , allows users to select a production *line*. When a new production *line* is added to the system, it will automatically appear in the list as shown below. No SQL statements, script or configuration is needed.

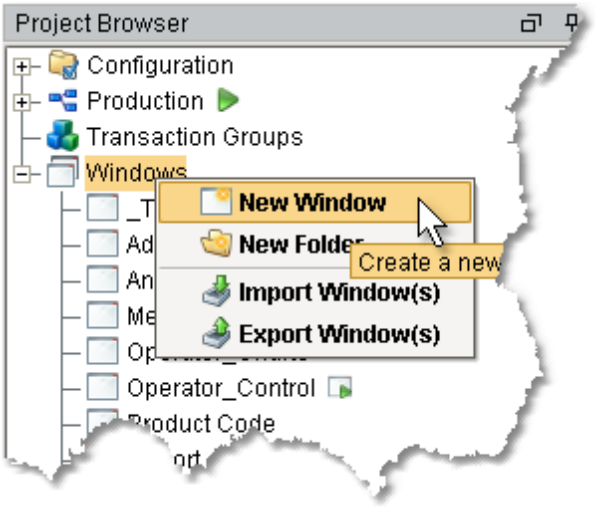


Production Line Selector Component

If the functionality of the components that come with OEE Downtime and Scheduling Module does not meet the project requirements, you can still use custom SQL statements, customer script and the standard Ignition components.

2.2.5.2 Creating a Screen

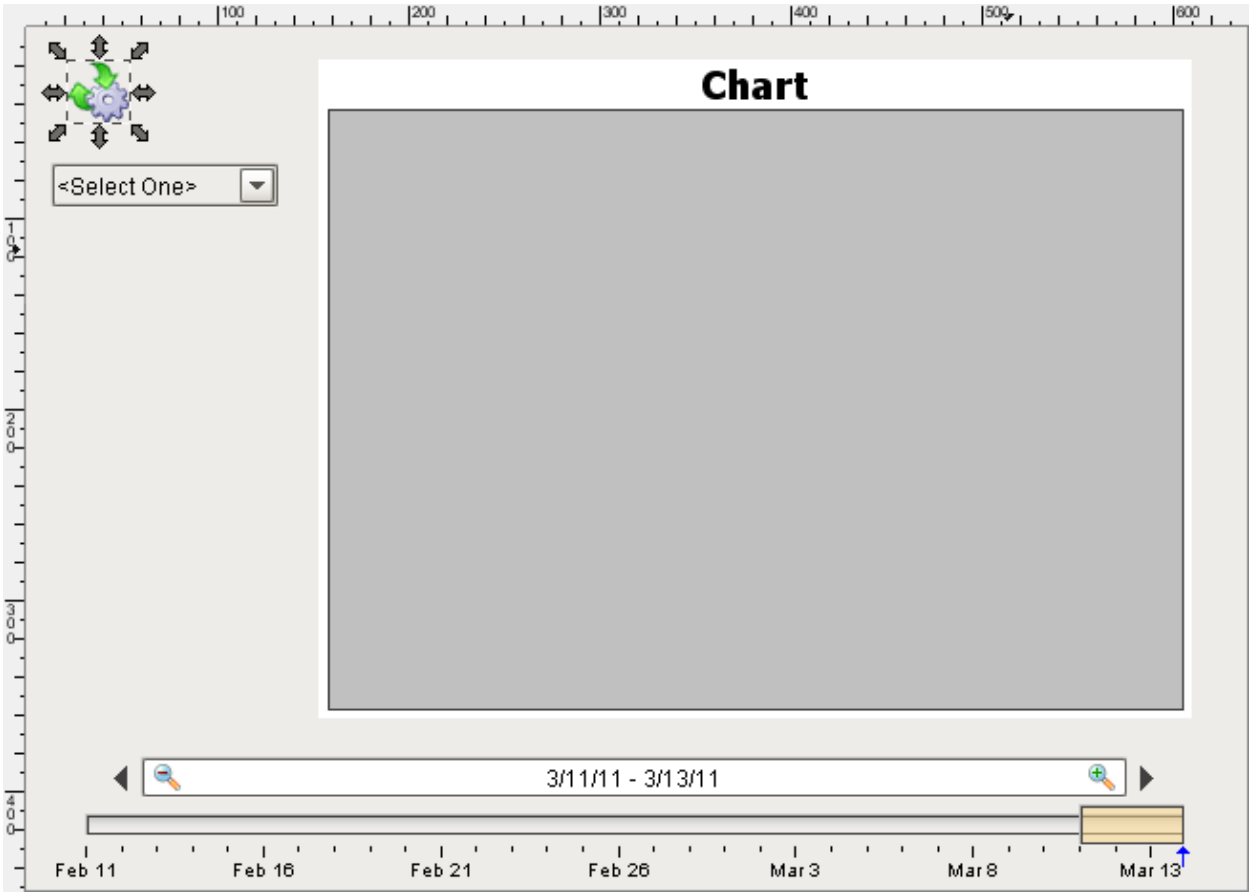
To add a new window in Ignition, right-click on the **Windows** node in the Project Browser and select the New Window menu item.



Adding a New Window

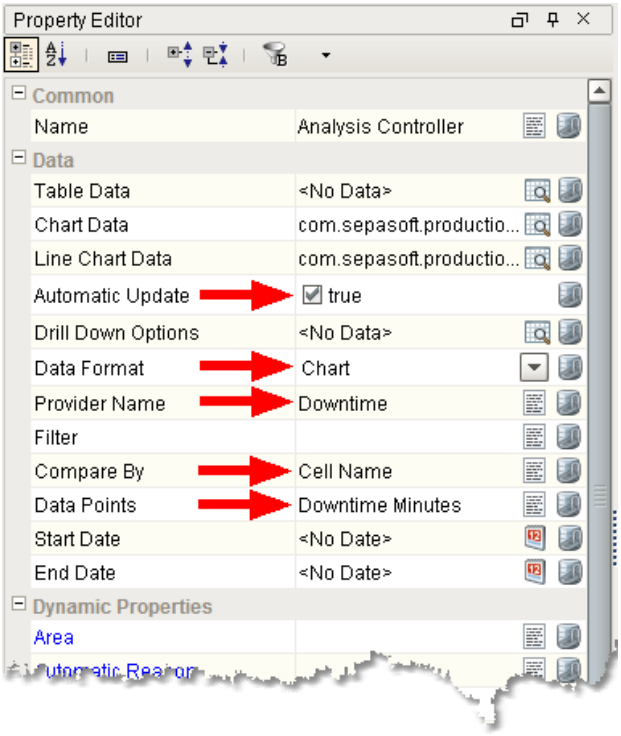
A new blank window will appear. Here is an example of a window you can create in Ignition.

First, drag a Analysis Controller, Production Line Selector, Production Bar Chart and Date Range components onto the new window as shown below.



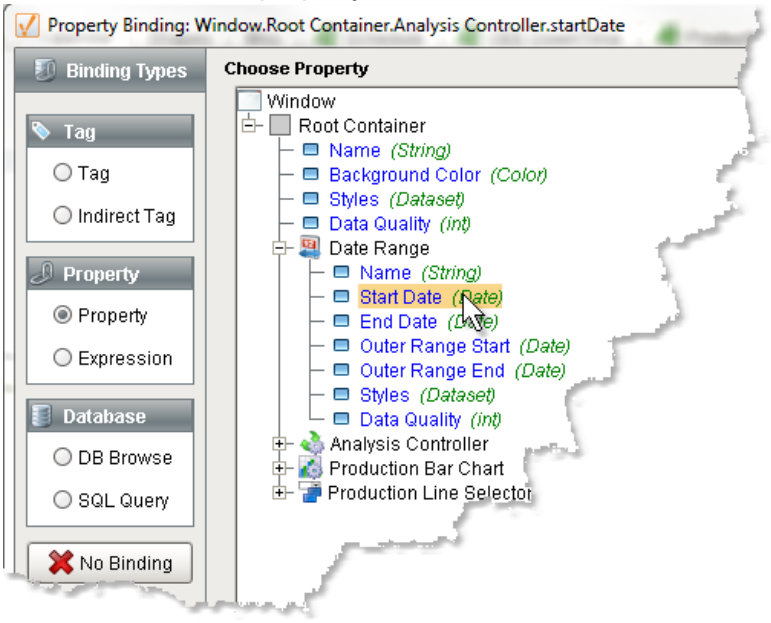
Example of a New Window

With the Analysis Controller selected, enter in the properties as shown below.



Property Editor

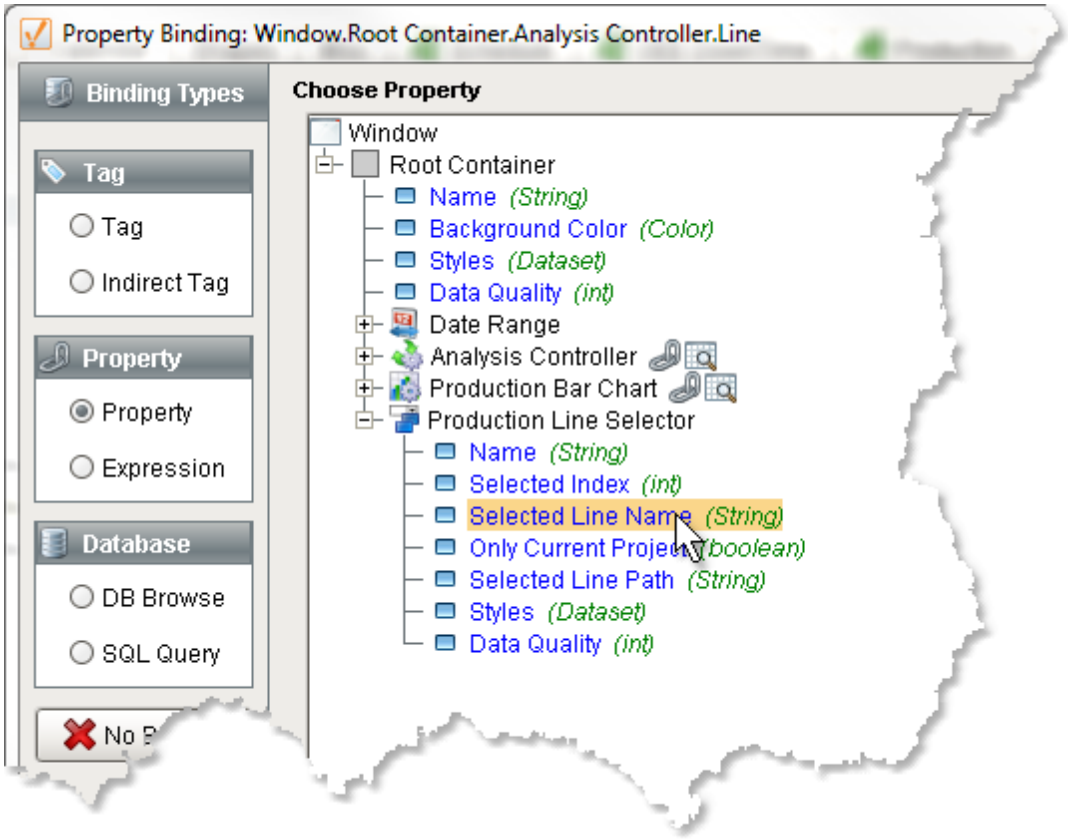
Now we will bind the date properties of the Analysis Controller component to the Date Range date properties. This will allow the user to select the date range that will affect the analysis results.
Do so by clicking the for the Start Date property, select **Property** binding type, navigate to the Date Range component and select Start Date property as shown below. Then click on the **OK** button.



Property Selector

Do the same for the End Date property, but select the End Date property of the Date Range component.

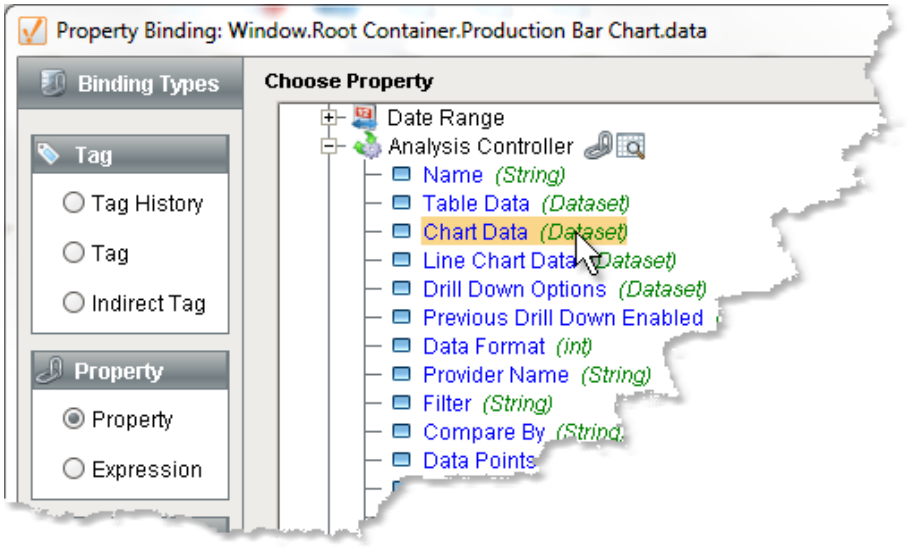
To allow the users to filter the analysis results by production *line*, we need to bind the Analysis Controller Line filter property to the Production Line Selector component.
Click on the for the Line property, select **Property** binding type, navigate to the Production Line Controller component and select Selected Line Name property as shown below. Click the **OK** button.



Property Selector

Now, click the Production Bar Chart on the new window. Next click the for the Data property, select **Property** binding type, navigate to the Analysis Controller component and select the Chart Data property as shown below. Click the **OK** button.

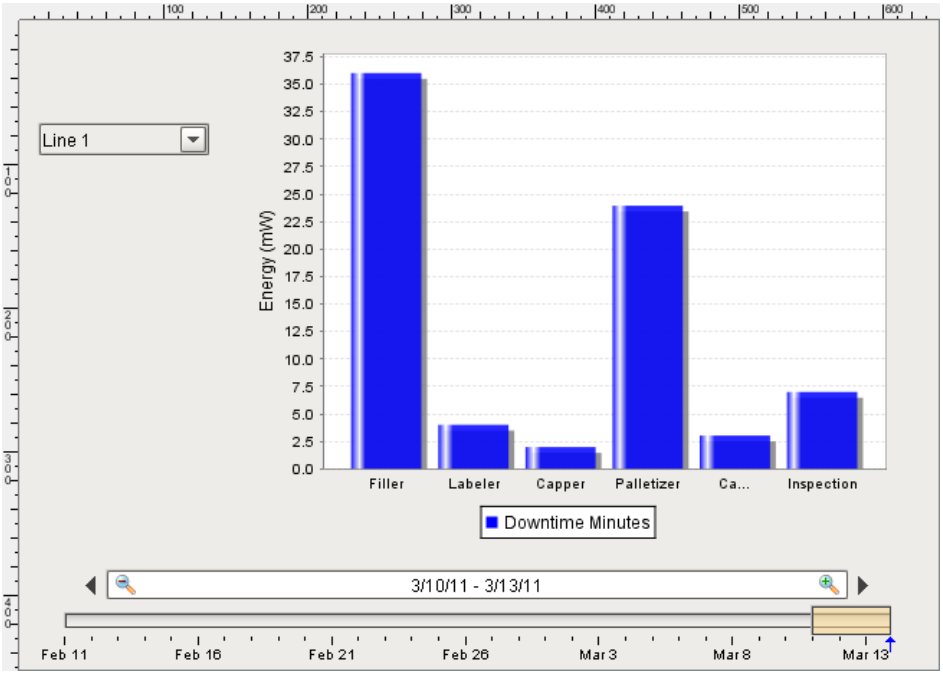
© Inductive Automation



Property Selector

To test, click on the preview button . This will allow use to use the screen as a user.

Select "Line 1" in the Production Line Selector component and you should see result as shown below.



Finished Example Window

Go ahead and play with the selected *line* and date range.

2.3 Configuration

There are two areas to configure the OEE Downtime and Scheduling module. The first area is in the Ignition Gateway and affects all MES Modules.

The second is in the Ignition Designer and is used to configure production models, user screens and the

like. These settings are saved in an Ignition project and can be backed up and restored using the built-in project backup and restore features of Ignition.

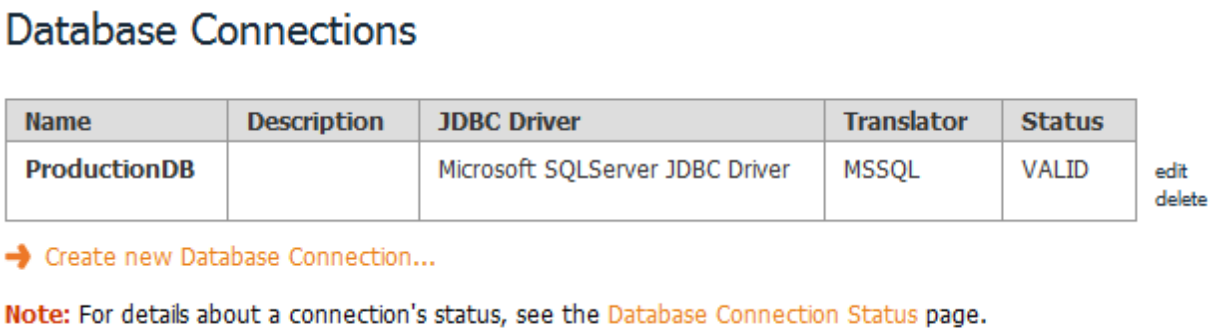
2.3.1 MES Module Configuration

The OEE, Downtime and Scheduling is just one of the MES (Manufacturing Execution System) modules that has settings which can be set.

2.3.1.1 Datasource Settings

OEE, downtime and schedule data is stored in databases external to Ignition. These database(s) are setup in the gateway configuration section by selecting the **Databases> Connections** section from the left-hand configuration menu in Ignition. See the Ignition documentation for more information on setting up a database connection.

Below shows a typical database connection that is required for the OEE, Downtime and Scheduling module.

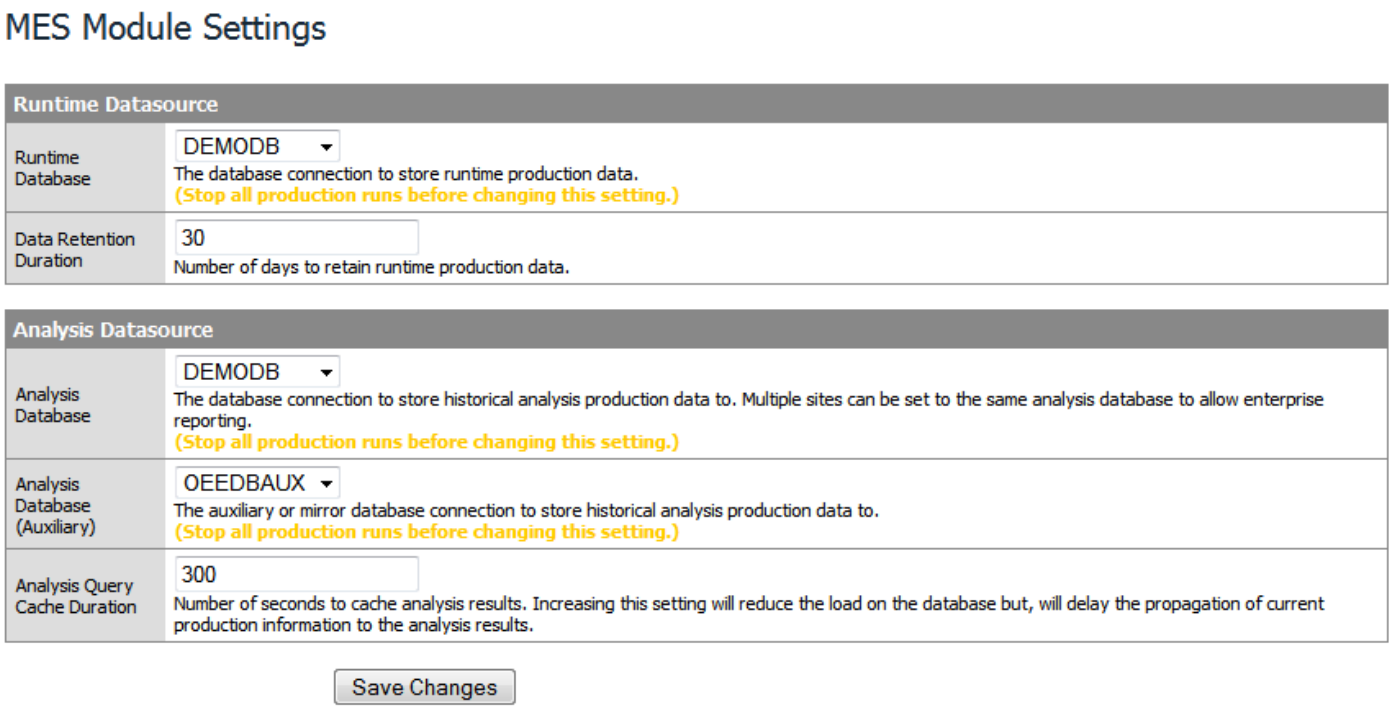


Sample Database Connection

To change the MES module settings, go to the configuration section in the gateway and select the **MES Modules> Settings** section from the left-hand side configuration menu.

Once a database connection is created, and if only one database connection exists, then it will be automatically selected to be used by the MES modules.
If more than one database connection exists, then the desired database connection can be selected to be used by the MES modules as shown below.

© Inductive Automation


MES Module Settings
Runtime Datasource
Runtime Database
DEMODB
The database connection to store runtime production data.
(Stop all production runs before changing this setting.)
Data Retention Duration
30
Number of days to retain runtime production data.
Analysis Datasource
Analysis Database
DEMODB
The database connection to store historical analysis production data to. Multiple sites can be set to the same analysis database to allow enterprise reporting.
(Stop all production runs before changing this setting.)
Analysis Database (Auxiliary)
OEEDBAUX
The auxiliary or mirror database connection to store historical analysis production data to.
(Stop all production runs before changing this setting.)
Analysis Query Cache Duration
300
Number of seconds to cache analysis results. Increasing this setting will reduce the load on the database but, will delay the propagation of current production information to the analysis results.
Save Changes

MES Module Settings Page

Runtime Database

The runtime database is where production and downtime data is stored during a production run. During a production run data is logged every minute or partial minute if a downtime event occurs, so a larger amount of data is stored in the runtime database.

Data Retention Duration

This setting specifies the number of days to retain the data in the runtime database after a production run has completed. The default setting is 30 days, This allows for viewing current and past production run information, down to the minute, for the past 30 days.

Analysis Database

The analysis database is where summarized production and downtime data is saved. For single production *site* installations, this can be the set to the same database as the runtime database. For multi-production *site* installations, all *sites* must set the analysis database to the same database to allow for *enterprise* analysis and reporting.

Analysis Database (Auxiliary)

The MES Modules will mirror the historical analysis data that is written to the local analysis database to this database. For single site implementations, set this to "-none-". For multi-site implementations, set this to the datasource for the common remote enterprise database.

Analysis Query Cache Duration

This setting represents the number of seconds to cache analysis results.

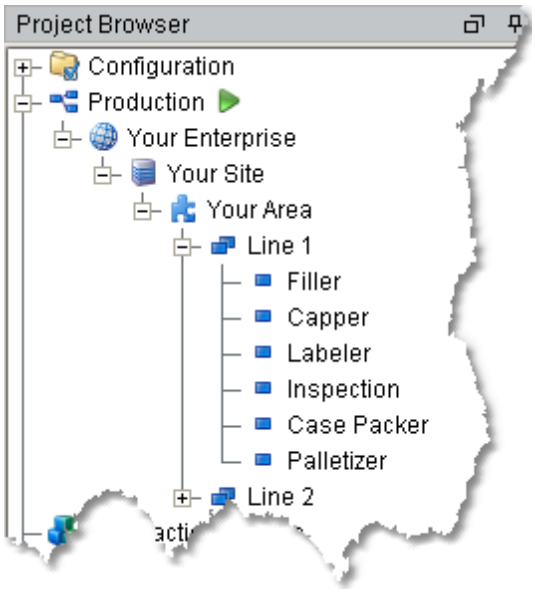
2.3.2 Production Model Configuration

A production model defines your manufacturing or process in tree view form. It provides an organized way to easily configure, control and analyze your facility. It starts with your *enterprise*, which represents your company, and continues down to the *site* (physical location), *area*, *line* and *cells*.

© Inductive Automation

2.3.2.1 Production Model

The production model is configured within the Ignition designer and is accessed by selecting the "Production" folder in the project browser. From here your *enterprise*, *site*, *area(s)*, *line(s)* and *cell(s)* can be added, renamed and deleted.



Production Model Tree

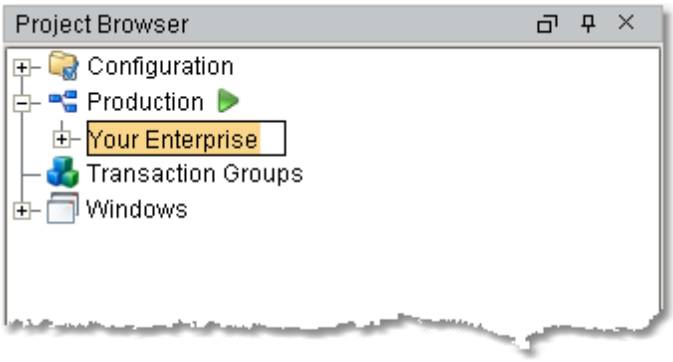
2.3.2.1.1 Enterprise Configuration

Adding an Enterprise

To add your *enterprise*, right-click on the "Production" folder in the project browser and select the **New Production Item > New Production Enterprise** menu item. An *enterprise* named "New Enterprise" will be added to the "Production" folder.

Renaming an Enterprise

To rename it to the name of your *enterprise*, right-click on it and select **Rename**, then enter the new name.



Enterprise Name

Deleting an Enterprise

To remove an existing *enterprise*, right-click on the *enterprise* item and select the **Delete** menu item. A window will appear confirming that you permanently want to delete the production *enterprise*. Please note

that the *site*, *area(s)*, *line(s)* and *cell(s)* underneath the *enterprise* will also be permanently removed.

© Inductive Automation

General Enterprise Settings

For the *enterprise*, there are only general settings. These settings are accessed by selecting the *enterprise* item contained in the ***"Production"*** folder in the project browser and then selecting the "General" tab as shown below.

Enterprise General Settings

Enabled — By default, added *enterprises* are enabled. It can be disabled by un-checking the Enabled setting and saving the project. This will stop the OEE, downtime and scheduling module from executing the *enterprise*, the *site* and all *area(s)*, *line(s)* and *cell(s)* that are underneath it.

Description — This is an optional description and is just for your reference.

2.3.2.1.2 Site Configuration

Adding a Site

To add your *site*, right-click on your *enterprise* folder in the project browser and select the **New Production Item > New Production Site** menu item. A *site* named "New Site" will be added to the *enterprise* folder.

Renaming a Site

To rename it to the name representing the *site's* physical location, right-click on it and select **Rename**, then enter the new name.

Deleting a Site

To remove an existing *site*, right-click on the *site* item and select the **Delete** menu item. A window will appear confirming that you permanently want to delete the production *site*. Please note that the *area(s)*, *line(s)* and cell(s) underneath the *site* will also be permanently removed.

© Inductive Automation

New Site

General Site Settings

These settings are accessed by selecting the *site* item contained in the *enterprise* folder in the project browser, and then selecting the "General" tab.

Enabled By default, added *sites* are enabled. It can be disabled by un-checking the Enabled setting and saving the project. This will stop the OEE, downtime and scheduling module from executing the *site* and all *area(s)*, *line(s)* and *cell(s)* that are underneath it.

Description This is an optional description and is just for your reference.

Shift 1:

Default Enabled If checked, shift 1 will be included during scheduling. If not checked, shift 1 will be scheduled around.

Default Start Time The time of day that first shift starts. The first shift ends at the start of second shift.

Shift 2:

Default Enabled If checked, shift 2 will be included during scheduling. If not checked, shift 2 will be scheduled around.

Default Start Time The time of day that second shift starts. The second shift ends at the start of third shift.

Shift 3:

Default Enabled If checked, shift 3 will be included during scheduling. If not checked, shift 3 will be scheduled around.

Default Start Time The time of day that third shift starts. The third shift ends at the start of first shift.

Note: The shift enabled and shift start times are the default for your production site and can be overridden by the production area and/or production line.

Schedule Settings

These settings are accessed by selecting the *site* item contained in the *enterprise* folder in the project browser and then selecting the "Schedule" tab as shown below. See Workday Routines for more information.

© Inductive Automation

Your Site

Site Production Item **Save project to make production changes active.**

General | OEE | Down Time | Schedule | Advanced

Workday Routine:

Name	Start Time	End Time
Grave Shift Night Break	1:00 AM	1:15 AM
Day Shift Meal Break	12:00 PM	12:30 PM
Swing Shift Morning Break	6:00 AM	6:15 AM
Swing Shift Meal Break	8:00 PM	8:30 PM
Day Shift Morning Break	9:00 AM	9:15 AM
Grave Shift Meal Break	3:00 AM	3:30 AM
Swing Shift Evening Break	5:00 PM	5:15 PM
Day Shift Afternoon Break	2:00 PM	2:15 PM
Swing Shift Night Break	10:00 PM	10:15 PM

Site Workday Routing List

Workday Routine Entry

See the Workday Routines section for more information.

Note: The workday routine entries are the default for your production site and can be overridden by the production area and/or production line.

2.3.2.1.3 Area Configuration

Adding an Area

To add a production *area*, right-click on your *site* folder in the project browser and select the **New Production Item > New Production Area** menu item. An *area* named "New Area" will be added to the *site* folder. Multiple production *areas* can be added to your production *site*. Each *area* can represent a physical or logical production *area* within your production *site*. Some examples of production *areas* are: packaging, cracking, filtration, fabrication, etc.

Renaming an Area

To rename it to the name representing the production *area*, right-click on it and select **Rename**, then enter the new name.

Deleting an Area

To remove an existing production *area*, right-click on the *area* item and select the **Delete** menu item. A window will appear confirming that you permanently want to delete the production *area*. Please note that the *line(s)* and *cell(s)* underneath the *area* will also be permanently removed.

New Area

© Inductive Automation

Area General Settings

These settings are accessed by selecting the desired *area* item contained in the *site* folder in the project browser and then selecting the "General" tab.

Enabled By default, added *areas* are enabled. It can be disabled by un-checking the Enabled setting and saving the project. This will stop the OEE, downtime and scheduling module from executing the *area* and all *line(s)* and *cell(s)* that are underneath it.

Description This is an optional description and is just for your reference.

Shift 1

Default Enabled If checked, shift 1 will be included during scheduling. If not checked, shift 1 will be scheduled around. To inherit the shift enabled from the from the *site*, select the "Inherit From Parent" option.

Default Start Time The time of day that first shift starts. The first shift ends at the start of second shift. To inherit the time of day that first shift starts setting from the *site*, select the "Inherit From Parent" option.

Shift 2

Default Enabled If checked, shift 2 will be included during scheduling. If not checked, shift 2 will be scheduled around. To inherit the shift enabled from the from the *site*, select the "Inherit From Parent" option.

Default Start Time The time of day that second shift starts. The second shift ends at the start of third shift. To inherit the time of day that second shift starts setting from the *site*, select the "Inherit From Parent" option.

Shift 3

Default Enabled If checked, shift 3 will be included during scheduling. If not checked, shift 3 will be scheduled around. To inherit the shift enabled from the from the *site*, select the "Inherit From Parent" option.

Default Start Time The time of day that third shift starts. The third shift ends at the start of first shift. To inherit the time of day that third shift starts setting from the *site*, select the "Inherit From Parent" option.

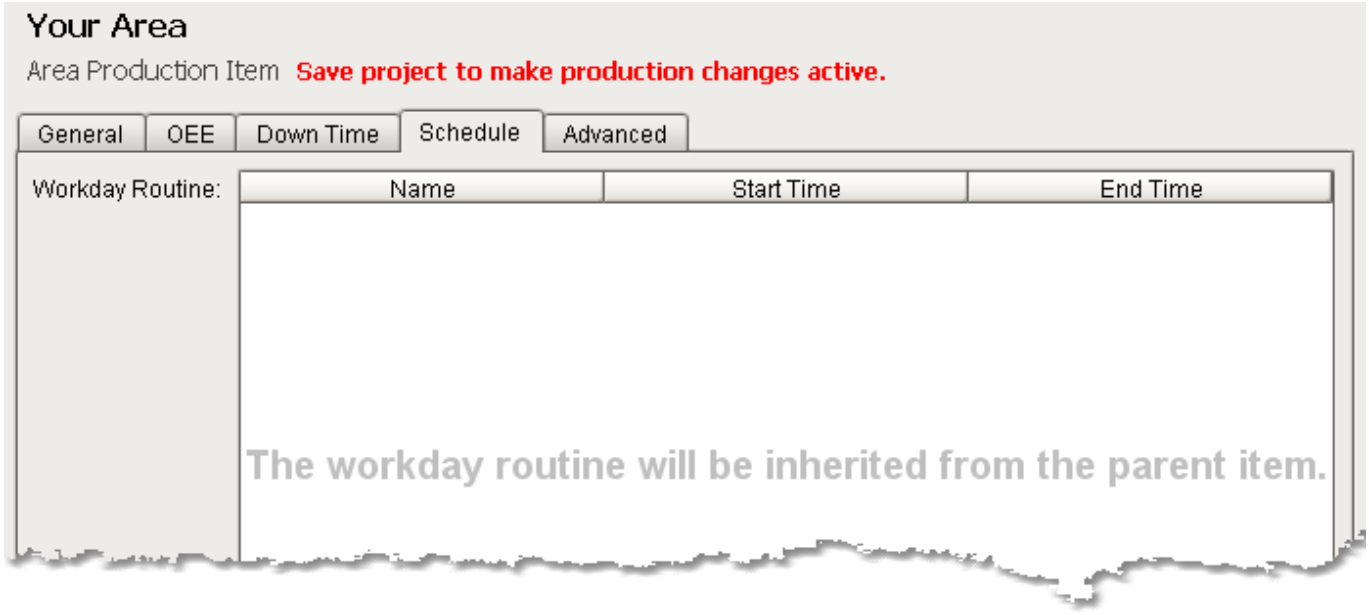
Note: The shift start times are the default for your production site and can be overridden by the production area and/or production line.

Area Schedule Settings

These settings are accessed by selecting the *area* item contained in the *site* folder in the project browser and then selecting the "Schedule" tab as shown below. See the Site Schedule Settings section for more information on workday routines.

If no *area* workday routine entries are entered, then they will be inherited from the production *site* as shown below.

© Inductive Automation



Area Workday Routine List

Workday Routine Entry
See the Workday Routines section for more information.

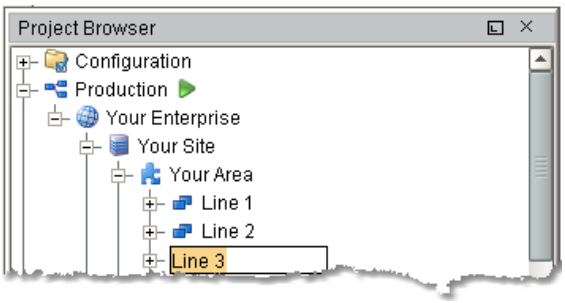
Note: The workday routine entries are the default for your production area and can be overridden by the production line.

2.3.2.1.4 Line Configuration

Adding a Line
To add a production *line*, right-click on an *area* folder in the project browser and select the **New Production Item > New Production Line** menu item. A *line* named "New Line" will be added to the *area* folder. Multiple production *lines* can be added to a production *area*.

Renaming a Line
To rename it to the name representing the production *line*, right-click on it and select **Rename**, then enter the new name.

Deleting a Line
To remove an existing production *line*, right-click on the *line* item and select the **Delete** menu item. A window will appear confirming that you permanently want to delete the production *line*. Please note that the cell(s) underneath the *line* will also be permanently removed.



New Line

Line General Settings

These settings are accessed by selecting the desired *line* item contained in the *area* folder in the project browser and then selecting the "General" tab.

Enabled By default, added *lines* are enabled. It can be disabled by un-checking the Enabled setting and saving the project. This will stop the OEE, downtime and scheduling module from executing the *line* and *cell(s)* that are underneath it.

Description This is an optional description and is just for your reference.

Shift 1

Default Enabled If checked, shift 1 will be included during scheduling. If not checked, shift 1 will be scheduled around. To inherit the shift enabled from the from the *area*, select the "Inherit From Parent" option.

Default Start Time The time of day that first shift starts. The first shift ends at the start of second shift. To inherit the time of day that first shift starts setting from the *area*, select the "Inherit From Parent" option.

Shift 2

Default Enabled If checked, shift 2 will be included during scheduling. If not checked, shift 2 will be scheduled around. To inherit the shift enabled from the from the *area*, select the "Inherit From Parent" option.

Default Start Time The time of day that second shift starts. The second shift ends at the start of third shift. To inherit the time of day that second shift starts setting from the *area*, select the "Inherit From Parent" option.

Shift 3

Default Enabled If checked, shift 3 will be included during scheduling. If not checked, shift 3 will be scheduled around. To inherit the shift enabled from the from the *area*, select the "Inherit From Parent" option.

Default Start Time The time of day that third shift starts. The third shift ends at the start of first shift. To inherit the time of day that third shift starts setting from the *area*, select the "Inherit From Parent" option.

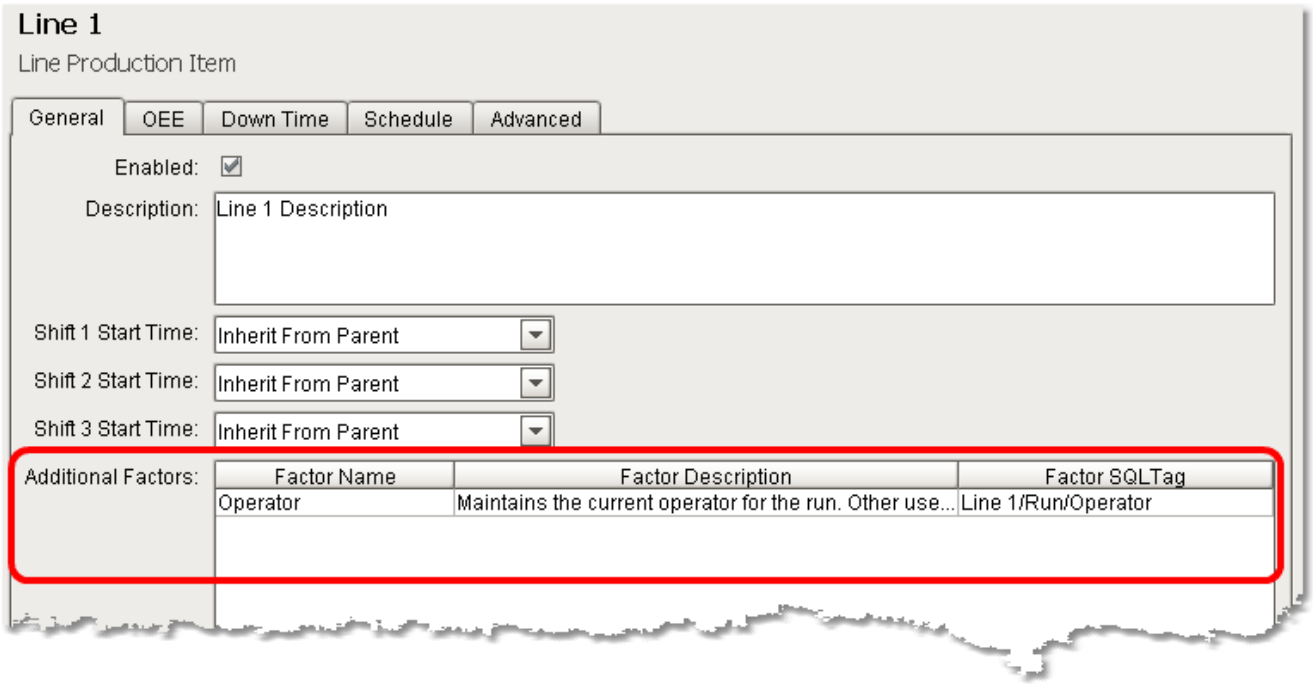
Additional Factors Additional Factors are user defined data points that are logged along with the production and downtime information. Once they are logged, they can be shown in charts, tables and reports. Additionally, other analysis can be done by filtering and/or setting up comparisons by their values.

Any value that can be read from an Ignition SQLTag can be added as a additional factor. This includes, values from barcode readers, databases, calculations, PLCs, or values derived from scripts, etc.

Example: An additional factor named cardboard manufacturer can be added. The operator can select the manufacturer that provided the cardboard or it can be obtained from some other source. Now, OEE and downtime results can be shown for each cardboard manufacturer. This can identify quality problems with raw material that directly affect efficiencies.

© Inductive Automation

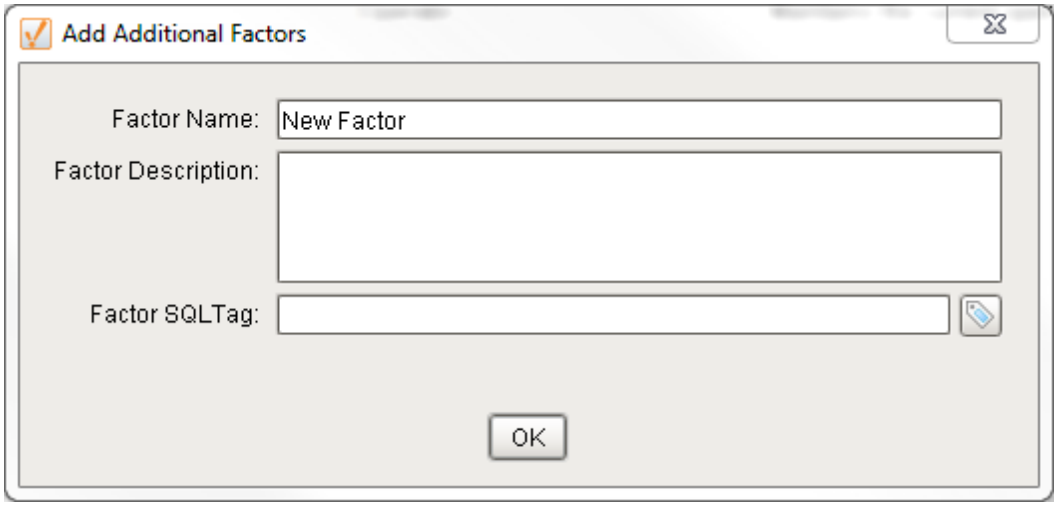
Below is an example of an operator additional factor. The operators name will be logged along with the production and downtime data. By doing so, OEE and downtime information can be filtered and grouped by the operator name.



Additional Factor List

Adding an Additional Factor

To add an additional factor, right-click anywhere on the additional factor table and select the **New** menu item. A dialog box will appear to allow entry of a new additional factor as shown below.



Additional Factor Settings

Factor Name

The required name of the additional factor is used to reference one additional factor from another. You can have any number of additional factors, but user usability will be hindered if too many are added. This is because the additional factors are added to user menus and if too many are added, the menus can become too long and confuse the end user.

© Inductive Automation

The name given to an additional factor should be meaningful to the end user. Again, this is because additional factors appear in menus allowing the end user to filter and group analysis and report data by them.

Factor Description

The optional description is just for reference or to keep internal notes about the additional factor.

Factor SQLTag

The required SQLTag is the source of the data value that will be logged. It is an Ignition SQLTag and the values can come from a PLC, a database query, other device in the field such as a barcode reader, expression, user input, or script. This opens the door to mesh any type of outside data into the MES module analysis and reporting.

Any type (format) of data that can be stored in an SQLTag can be logged. If SQLTag value is a string, then the end user can filter and group by the additional factor. If the SQLTag is a number, the option to filter and group by the additional factor will not be shown to the end user.

The SQLTag can be manually typed or pasted into the Factor SQLTag edit box. Optionally, clicking on the icon will display a browser where a SQLTag can be selected.

Editing an Additional Factor

To edit an existing additional factor, right-click on the desired entry in the additional factor table and select the **Edit** menu item. A dialog box similar to the add dialog box will appear, allowing editing of the additional factor.

Deleting an Additional Factor

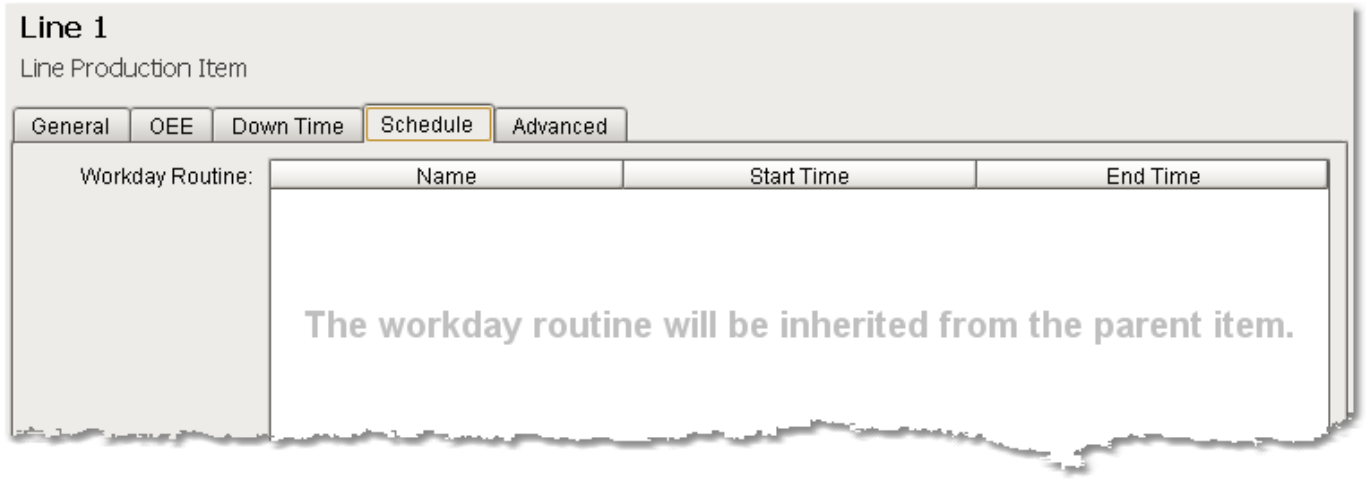
To remove an existing additional factor entry, right-click on the desired entry in the additional factor table and select the **Delete** menu item. A window will appear confirming that you want to remove the additional factor. The additional factor will no longer be logged. However, any production runs that occurred before the additional factor was deleted, will still show in the analysis and reporting.

Line Schedule Settings

These settings are accessed by selecting the *line* item contained in the *area* folder in the project browser and then selecting the "Schedule" tab as shown below. See the Site Schedule Settings section for more information on workday routines.

If no *area* workday routine entries are entered, then they will be inherited from the production *area* as shown below.

© Inductive Automation



Line Workday Routine List

Workday Routine Entry

See the Workday Routines section for more information.

Other Line Schedule Settings

Default Schedule Rate	This default production rate used for scheduling purposes. Because the standard production rate is typically not achieved, a scheduling rate is used when determining the work order finish time during scheduling. The actual scheduling rate used is determined from the product code and *line* that is being scheduled.
Schedule Rate Period	The period of time used for the scheduling rate. The options are Hour and Minute.
Auto Start Schedule Entries	If true, the scheduled entries on the calendar will automatically start at the scheduled time. If false, scheduled entries can be chosen out of order and started manually, typically by the operator clicking the **Start** button.
Auto Start Production After Changeover	Determines the behaviour when the change over time has expired. If true, the production run will automatically start. If the *line* is not running, then downtime will start being accumulated. If false, the production run must be started by some other means. Typically, this is done by the operator clicking the **Start** button but it can be accomplished by programmatically setting the *Enable Run* property for the *line*.

© Inductive Automation

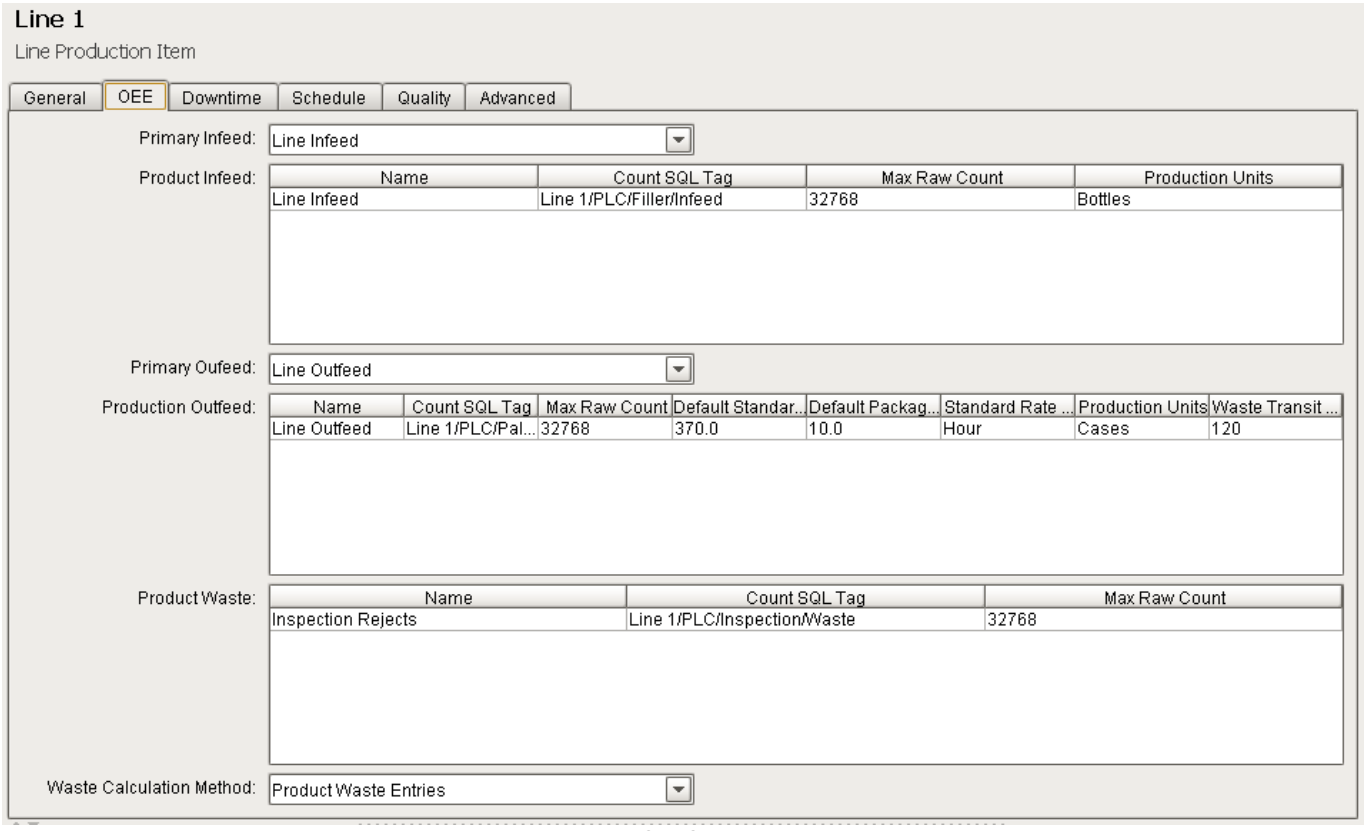
Line OEE Settings

The Line OEE settings are accessed by selecting the *line* item contained in the *area* folder in the project browser, and then selecting the "OEE" tab as shown below.

Before OEE calculations can be performed, production count information is required. At a minimum, the outfeed production count for a production *line* is needed. Additional production count information can be configured, which will result in more OEE calculations. For example, if the infeed production count is configured for a production, then product accumulation and waste can also be calculated. Also, OEE Performance uses items started vs. standard rate so that it is isolated from quality factors. When the infeed production count is not used and quality is being used, then quality will not be isolated from performance.

If a production *line* is configured for more than one infeed or outfeed, then accumulation and waste calculations will be performed for each combination. For example, a production *line* can be configured to track container, caps and product as infeeds, and a single outfeed of full containers. The independent waste calculations for containers, caps and production will be performed. See Production Count Tracking section for more information.

Below is an example showing a single infeed and outfeed configure for a production *line*.



Line OEE Settings

Primary Infeed

The production *line* OEE waste is derived from the primary infeed. If a production *line* has been configured for multiple infeeds, select the infeed that is to be used for the waste calculation.

Product Infeeds

For each infeed, the OEE module will start calculating production rate per minute or production rate per

hour values. These values can be accessed through the Production OPC Server. See the section on Production OPC Values, and the section on Product Infeed for more information.

Adding a Product Infeed
See the section on Adding a Product Infeed for details on adding product infeed entries.

Editing a Product Infeed
See the section on Editing a Product Infeed for details on editing product infeed entries.

Deleting a Product Infeed
See the section on Deleting a Product Infeed for details on deleting product infeed entries.

Primary Outfeed
The production *line* OEE waste is derived from the primary outfeed. If a production *line* has been configured for multiple outfeeds, select the outfeed that is to be used for the waste calculation.

Product Outfeeds
For each outfeed, the OEE module will start calculating production rate per minute, or production rate per hour values. These values can be accessed through the Production OPC Server. See the section on Production OPC Values for more information. See the section on Product Outfeed for more information.

Adding a Product Outfeed
See the section on Adding a Product Outfeed for details on adding product outfeed entries.

Editing a Product Outfeed
See the section on Editing a Product Outfeed for details on editing product outfeed entries.

Deleting a Product Outfeed
See the section on Deleting a Product Outfeed for details on deleting product outfeed entries.

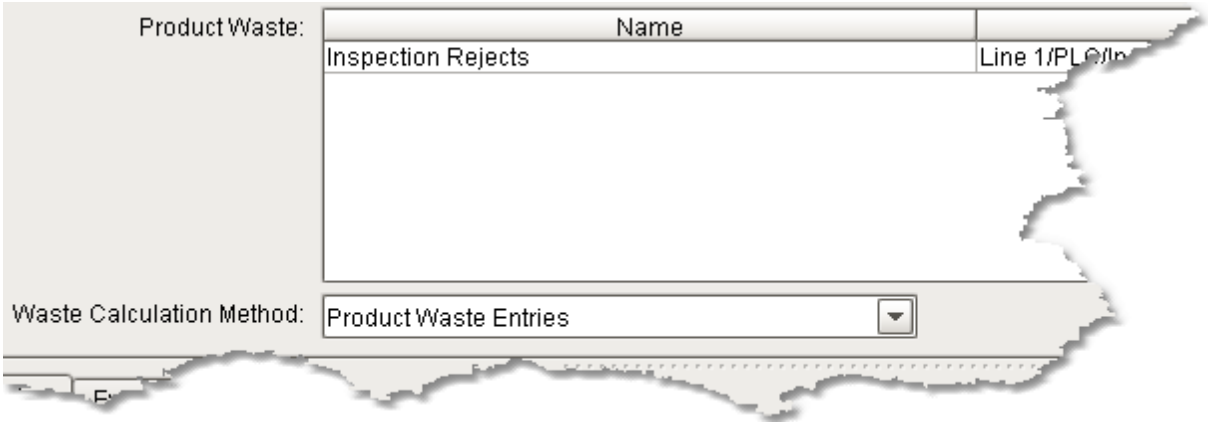
Product Waste
For each waste entry, the OEE module will start tracking true waste count values. These values can be accessed through the Production OPC Server. See the section on Production OPC Values for more information. See the section on Product Waste for more information.

Adding a Product Waste Entry
See the section on Adding a Product Waste Counter for details on adding product waste entries.

Editing a Product Outfeed
See the section on Editing a Product Waste Counter for details on editing product waste entries.

Deleting a Product Outfeed
See the section on Deleting a Product Waste Counter for details on deleting product waste entries.

© Inductive Automation



Waste Calculation Methods

Waste Calculation Methods

There are several ways to calculate the amount of waste coming from a production line:

None

No waste calculation methods will be used to determine waste counts.

Run Waste Count Tag

Waste will be calculated based on what is entered in the Run Waste Count tag. This may be a unique formula used to calculate waste or a manual entry of the waste count.

Started vs. Finished

When this method is used, outfeed will be compared to infeed to determine how many units were lost to waste. If 1000 units enter the production line, and 900 finished products exit the production line, then it is assumed that 100 units were lost to waste. A Waste Transit time other than 0 must be entered for the primary outfeed. The Waste Transit time should be the normal amount of time a production unit takes to travel from infeed (started) to outfeed (finished).

Product Waste Entries

In order to calculate waste by this method, one or more entries must be in the Product Waste table. The waste from each entry is added together to equal the total amount of waste. For example, there may be two cells on the line that inspect and discard faulty products. If the first cell discarded 10 units, and the second cell discarded 5 units, the total waste count would be 15 units.

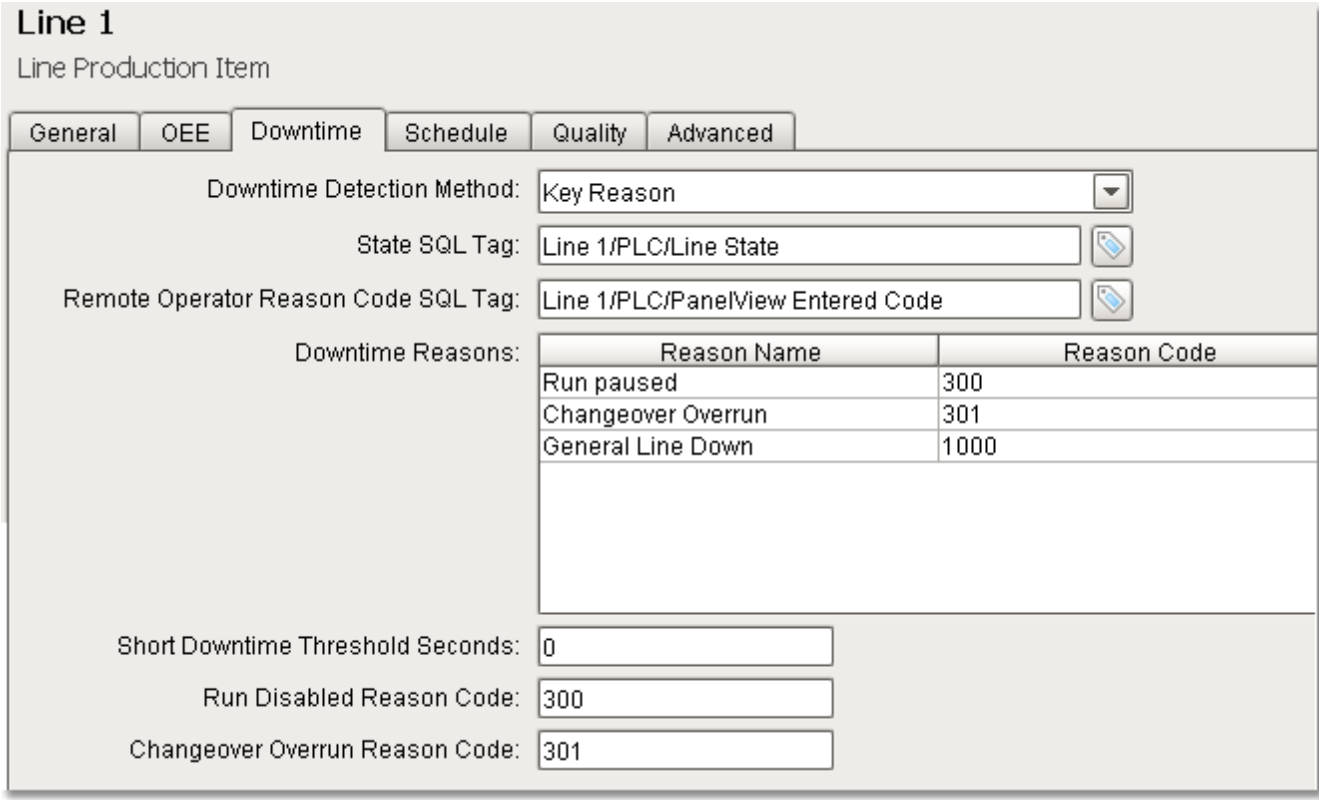
Sum Product Waste From Each Cell

This method will sum up each of the cell's waste values. Product Waste Entries will be ignored.

Line Downtime Settings

These settings are accessed by selecting the *line* item contained in the *site* folder in the project browser and then selecting the "Downtime" tab as shown below. Once downtime reasons have be added, the OEE, Downtime and Scheduling module will either check the list if the *line* stops running or allow the operator to select reason. See the section on Downtime Reasons for more information.

© Inductive Automation



Line Downtime Settings

Downtime Detection Method

To determine the reason a production *line* or process is down, set the Downtime Detection Method setting:

1. Select ***Initial Reason*** to select the initial *cell* that is down as the reason the *line* is down. If there is a *cell group* on the *line* and the Minimum Cells Running Threshold has not been reached for that *cell group*, the first *cell* with recordable downtime within the *cell group* is the reason that the *line* is down. If the Minimum Cells Running Threshold has been reached, no part of the *cell group* will be recorded as the cause of downtime, and the next *cell* or *cell group* will be examined.
2. Select ***Key Reason*** to select the first cell, in the order they appear in the designer, that is down as the reason the *line* is down. The line state will be checked first, overriding the cell state if the *line* is already down. If there is a *cell group* on the *line* and the Minimum Cells Running Threshold has not been reached for that *cell group*, the last *cell* within the *cell group* that guarantees the threshold will not be reached with recordable downtime is the reason that the *line* is down. If the Minimum Cells Running Threshold has been reached, no part of the *cell group* will be recorded as the cause of downtime, and the next *cell* or *cell group* will be examined.
3. Select ***Line State*** to ignore the *cells* and use the value of the State SQLTag that is configured for the *line*.

See the section on Downtime Reasons for more information on each method.

State SQLTag

This SQLTag is to read the current state of the *line* or process. Note that if this tag is set, then if its current state is not 1 it will supersede Key Reason or Initial Reason methods states.

It is an Ignition SQLTag and the values can come from a PLC, a database query, other device in the field such as a barcode reader, an expression, user input, or script.

© Inductive Automation

The data type (format) of the SQLTag containing the state must be a number. The SQLTag can be manually typed or pasted in to the Factor SQLTag edit box. Optionally, clicking on the icon will display a browser where a SQLTag can be selected.

Remote Operator Reason Code SQLTag

This code will allow changing of the active downtime Operator reason code from a tag instead of directly in the downtime table. It is up to you to determine the proper code to write to the tag. If the code is not valid then the code will not be changed. For instance, if the current active code is from the "Capper" cell of the line then the remote code must be an operator selectable code from the "Capper".

Downtime Reasons

Adding a Downtime Reason

See the section on Adding a Downtime Reason for details on adding downtime reason entries.

Editing a Downtime Reason

See the section on Editing a Downtime Reason for details on editing downtime reason entries.

Deleting a Downtime Reason

See the section on Deleting a Downtime Reason for details on deleting downtime reason entries.

Short Downtime Threshold Seconds

Short downtime are events that last a small specified time, 120 seconds for instance. Short events will not affect the OEE availability calculation. If set to 0 then all downtime events are considered long and will always affect the OEE availability calculation. This setting affects this Line and all Cells associated to this Line. Short downtime events are always recorded and can be displayed via the analysis components.

Run Disabled Reason Code

Anytime a production run is ended and then later resumed, this reason code will be used as a downtime reason. A downtime reason with the same reason code must exist in the downtime reason table. The reason can be set to planned or unplanned downtime to produce the desired results during analysis and reporting.

Changeover Time Reason Code

When changeover time is scheduled for a production run, but production does not begin when the changeover ends, this reason code will be used as a downtime reason. A downtime reason with the same reason code must exist in the downtime reason table. The reason can be set to planned or unplanned downtime to produce the desired results during analysis and reporting.

See the section on Downtime Reasons for more information.

2.3.2.1.5 Cell Group Configuration

Adding a Cell Group

To add a production cell group, right-click on a *line* folder in the project browser and select the **New Production Item > New Production Cell Group** menu item. A cell group named "New Cell Group" will be added to the *line* folder. Multiple production cell groups can be added to a production *line*.

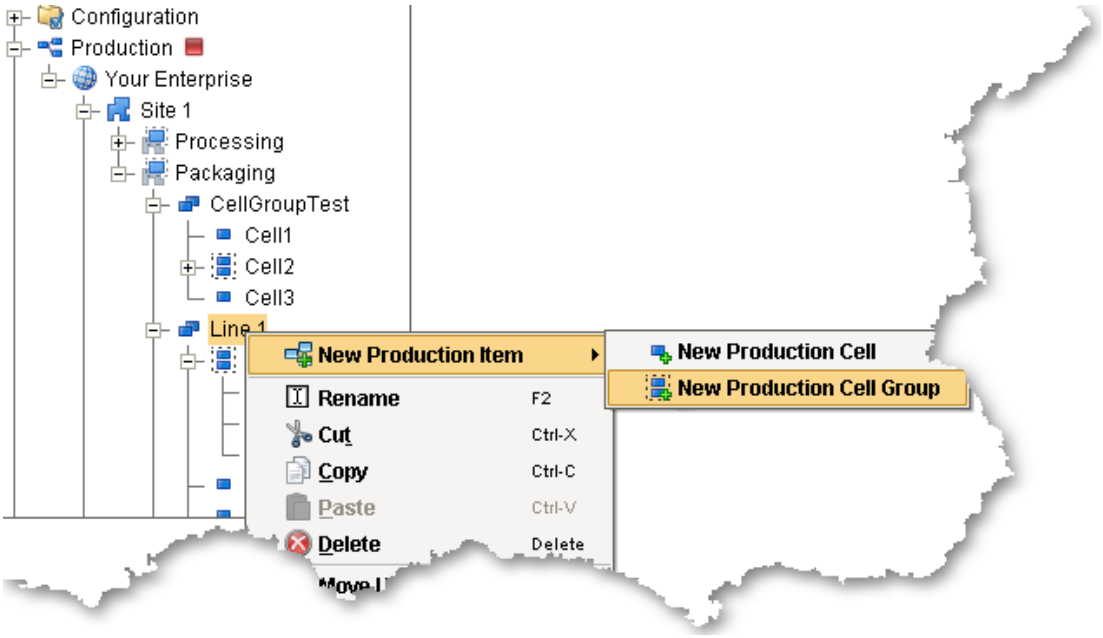
Renaming a Cell Group

To rename it to the name representing the production cell group, right-click on it and select **Rename**, then enter the new name.

© Inductive Automation

Deleting a Cell Group

To remove an existing production cell group, right-click on the cell group item and select the **Delete** menu item. A window will appear confirming that you permanently want to delete the production cell group.



Adding a Cell Group

Cell Group General Settings

These settings are accessed by selecting the desired cell group item contained in the *line* folder in the project browser and then selecting the "General" tab.

Enabled — By default, added cell groups are enabled. It can be disabled by un-checking the Enabled setting and saving the project. This will stop the OEE, downtime and scheduling module from executing the cell group.

Description — This is an optional description and is just for your reference.

Cell Group OEE

There are no additional settings for *cell group* OEE; however, the OEE of the *cell group* is calculated by averaging the OEE and Production Counts of all the *cells* within the group.

Cell Group Downtime Settings

Minimum Cells Running Threshold: This is the minimum number of *cells* that must be running within the *cell group* in order for the *cell group* to be considered running. If there are five *cells* within a *cell group*, and the Minimum Cells Running Threshold is 3, then three *cells* must be running. If two out of the five *cells* are down, there are still three *cells* running, so the *cell group* is running. If three out of the five *cells* are down, there are only two *cells* running, meaning the *cell group* is down because the threshold has not been met.

© Inductive Automation

2.3.2.1.6 Cell Configuration

Adding a Cell

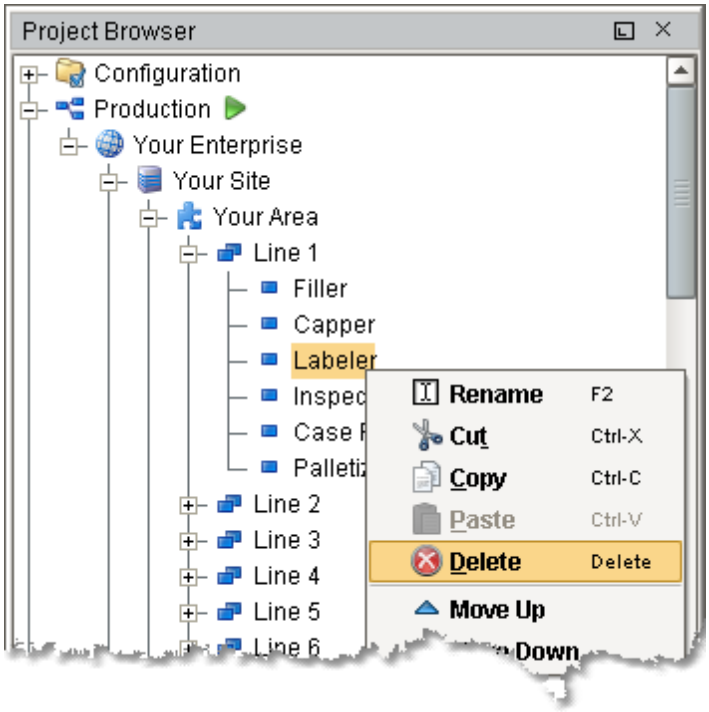
To add a production cell, right-click on a *line* folder in the project browser and select the **New Production Item > New Production Cell** menu item. A cell named "New Cell" will be added to the *line* folder. Multiple production cells can be added to a production *line*.

Renaming a Cell

To rename it to the name representing the production cell, right-click on it and select **Rename**, then enter the new name.

Deleting a Cell

To remove an existing production cell, right-click on the cell item and select the **Delete** menu item. A window will appear confirming that you permanently want to delete the production cell.



Deleting a Cell

Cell General Settings

These settings are accessed by selecting the desired cell item contained in the *line* folder in the project browser and then selecting the "General" tab.

Enabled By default, added cells are enabled. It can be disabled by un-checking the Enabled setting and saving the project. This will stop the OEE, downtime and scheduling module from executing the cell.

Description This is an optional description and is just for your reference.

Cell OEE Settings

The Cell OEE settings are accessed by selecting the cell item contained in the *line* folder in the project

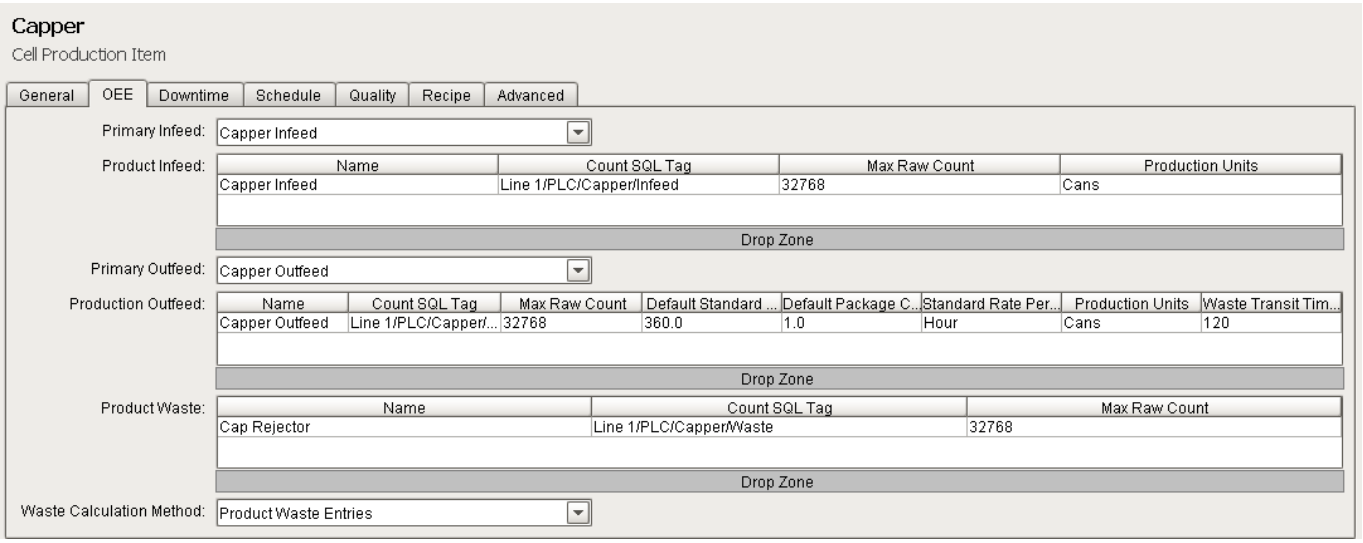
browser and then selecting the "OEE" tab as shown below.

For production cells, the OEE settings are optional and are only needed if you want to track efficiencies, waste or monitor production rate by individual production cells. It is also important to note that the OEE information is not required to track downtime for the cell.

Before OEE calculations can be performed, production count information is required. At a minimum, the outfeed production count for a production cell is needed if tracking OEE for it is desired . Additional production count information can be configured, which will result in more OEE calculations. For example, if the infeed production count is configured for a production, then product accumulation and waste is calculated.

If a production cell is configured for more than one infeed or outfeed, then accumulation and waste calculations will be performed for each combination. For example, a production cell can be configured to track containers and caps as infeeds and a single outfeed of full containers. The independent waste calculations for containers and caps will be performed. See Production Count Tracking section for more information.

Below is an example showing a single infeed and outfeed configure for a production cell.



Cell OEE Settings

Primary Infeed

The production cell OEE waste is derived from the primary infeed. If a production cell has been configured for multiple infeeds, select the infeed that is to be used for the waste calculation.

Product Infeeds

For each infeed, the OEE module will start calculating production rate per minute, production rate per hour values. These values can be accessed through the Production OPC Server. See the section on Production OPC Values for more information. For the section on Product Infeed for more information.

Adding a Product Infeed

See the section on Adding a Product Infeed for details on adding product infeed entries.

Editing a Product Infeed

See the section on Editing a Product Infeed for details on editing product infeed entries.

Deleting a Product Infeed

See the section on Deleting a Product Infeed for details on deleting product infeed entries.

Primary Outfeed

The production *line* OEE waste is derived from the primary outfeed. If a production cell has been configured for multiple outfeeds, select the outfeed that is to be used for the waste calculation.

Product Outfeeds

For each outfeed, the OEE module will start calculating production rate per minute, or production rate per hour values. These values can be accessed through the Production OPC Server. See the section on Production OPC Values, and the section on Product Outfeed for more information.

Adding a Product Outfeed

See the section on Adding a Product Outfeed for details on adding product outfeed entries.

Editing a Product Outfeed

See the section on Editing a Product Outfeed for details on editing product outfeed entries.

Deleting a Product Outfeed

See the section on Deleting a Product Outfeed for details on deleting product outfeed entries.

Product Waste

For each waste entry, the OEE module will start tracking true waste count values. These values can be accessed through the Production OPC Server. See the section on Production OPC Values, and the section on Product Waste for more information.

Adding a Product Waste Entry

See the section on Adding a Product Waste Counter for details on adding product waste entries.

Editing a Product Outfeed

See the section on Editing a Product Waste Counter for details on editing product waste entries.

Deleting a Product Outfeed

See the section on Deleting a Product Waste Counter for details on deleting product waste entries.

Waste Calculation Methods

There are several ways to calculate the amount of waste coming from a production cell:

None

No waste calculation methods will be used to determine waste counts.

© Inductive Automation

Run Waste Count Tag

Waste will be calculated based on what is entered in the Run Waste Count tag. This may be a unique formula used to calculate waste or a manual entry of the waste count.

Started vs. Finished

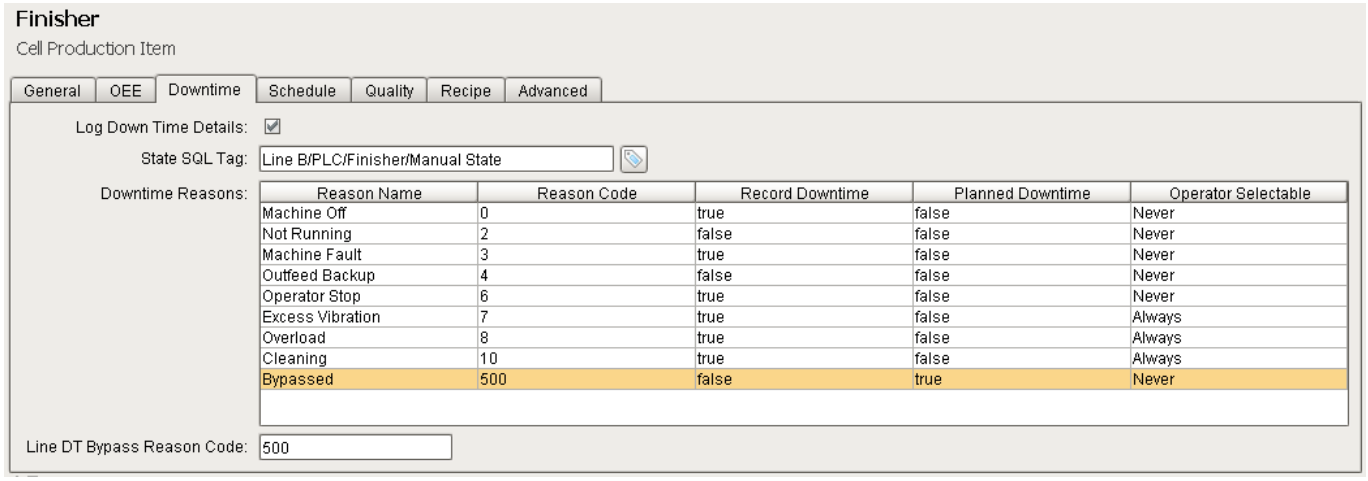
When this method is used, outfeed will be compared to infeed to determine how many units were lost to waste. If 1000 units enter the production cell, and 900 finished products exit the production cell, then it is assumed that 100 units were lost to waste. A Waste Transit time other than 0 must be entered for the primary outfeed. The Waste Transit time should be the normal amount of time a production unit takes to travel from infeed (started) to outfeed (finished).

Product Waste Entries

In order to calculate waste by this method, one or more entries must be in the Product Waste table. The waste from each entry is added together to equal the total amount of waste. For example, there may be two points on the cell that inspect and discard faulty products. If the first point discarded 10 units, and the second point discarded 5 units, the total waste count would be 15 units.

Cell Downtime Settings

These settings are accessed by selecting the *line* item contained in the *site* folder in the project browser and then selecting the "Downtime" tab as shown below. Once downtime reasons have be added, the OEE, Downtime and Scheduling module will either check the list if the *line* stops running or allow the operator to select the reason. See the section on Downtime Reasons for more information.



Reason Name	Reason Code	Record Downtime	Planned Downtime	Operator Selectable
Machine Off	0	true	false	Never
Not Running	2	false	false	Never
Machine Fault	3	true	false	Never
Outfeed Backup	4	false	false	Never
Operator Stop	6	true	false	Never
Excess Vibration	7	true	false	Always
Overload	8	true	false	Always
Cleaning	10	true	false	Always
Bypassed	500	false	true	Never

Cell Downtime Settings

Log Downtime Details

Cell downtime logging is independent from *line* downtime. To log all of the downtime details for a cell, check the Log Downtime Details setting. This will cause all downtime events for the cell to be logged to the database. If this amount of detail is not used, it is recommended to uncheck this setting as it saves space in the database.

See Downtime Reasons for more information.

State SQLTag

This is the SQLTag used to read the current state of the cell. It is an Ignition SQLTag and the values can come from a PLC, a database query, other device in the field such as a barcode reader, an expression,

user input, or script.

The data type (format) of the SQLTag containing the state must be a number.
The SQLTag can be manually typed or pasted in to the Factor SQLTag edit box. Optionally, clicking on the icon will display a browser where a SQLTag can be selected.

Downtime Reasons

Adding a Downtime Reason
See the section on Adding a Downtime Reason for details on adding downtime reason entries.

Editing a Downtime Reason
See the section on Editing a Downtime Reason for details on editing downtime reason entries.

Deleting a Downtime Reason
See the section on Deleting a Downtime Reason for details on deleting downtime reason entries.

Line DT Bypass Reason Code
When a cell is in bypass, this is the reason code that will be recorded. It will override the actual code that is returned in the State SQL Tag. This allows any reports or displays to show the desired state while the cell is bypassed.
A cell may be placed in bypass at runtime by selecting the "Line DT Bypass" Production OPC tag for this cell.

2.3.3 Workday Routines

Workday routine activities can be breaks, lunches, safety meetings or anything that is scheduled, non-production times that occur every day.

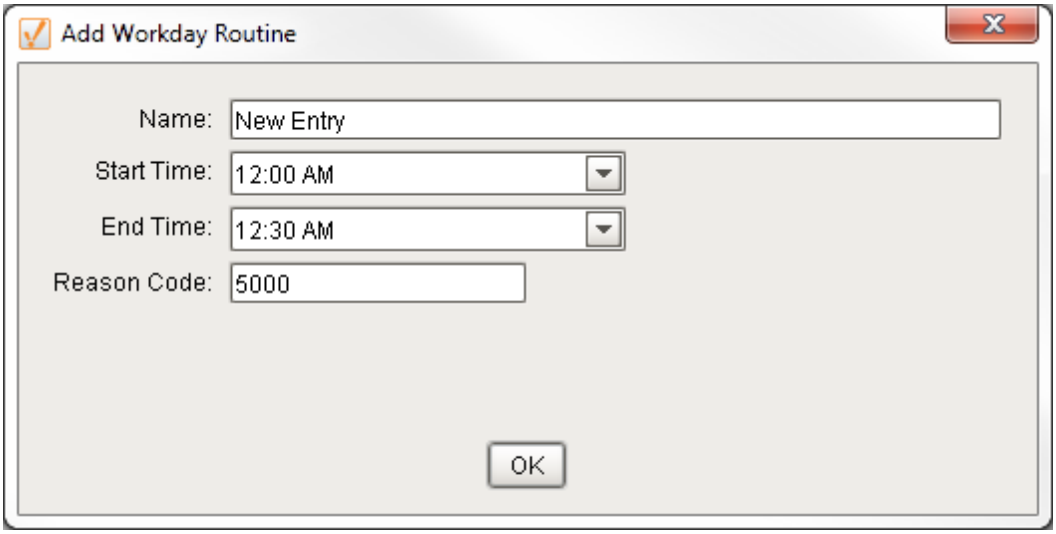
Scheduling
When production runs are scheduled by the production planner, these workday routine items are scheduled around.
For example if you schedule a run and it would take 4 hours to produce the scheduled quantity, a 30 minute workday routine will extend a schedule end time by 30 minutes if the schedule falls within the routines start and end times.

Line Running
If no reason code is entered or it does not match a downtime reason code for the line then it will be ignored.

If a reason code is entered that matches a downtime reason for the line then that reason will be triggered at the start time and ended at the end time. When the reason code is defined as a planned downtime then the time will not count against the OEE of the production run.
NOTE: A matched reason code will override any other downtime events that occur on the line during the defined time.

Adding a Workday Routine
To add a workday routine entry, right-click anywhere on the table containing workday routines and select the **New** menu item. A dialog box will appear to allow entry of a name, start time, end time and optional reason code for the workday routine entry as shown below.

© Inductive Automation



Workday Routine Entry Settings

Editing a Workday Routine

To edit an existing workday routine entry, right-click on the desired entry in the workday routine table and select the **Edit** menu item. A dialog box similar to the add dialog box will appear allowing editing of the entry.

Deleting a Workday Routine

To remove an existing workday routine entry, right-click on the desired entry in the workday routine table and select the **Delete** menu item. A window will appear confirming that you want to remove the workday routine entry.

Import/ Export

To export workday routine entries, right-click anywhere on the table containing workday routines and select the **Export** menu item. A dialog box will appear to allow selection of an existing file or the typing in of a name of the new file to save the workday routine entries to. If a file extension is not entered, then the default .csv will be used.

To import workday routine entries, right-click anywhere on the workday routine table and select the **Import** menu item. A dialog box will appear to allow selection of a comma separated values (csv) formatted file.

The first *line* of the file must at least contain the property names separated by commas. If additional names exist, they will be ignored. The property names can be in any order. Below is a sample csv file showing multiple workday routine entries.

```
Name,Start Time,End Time,Reason Code
"Graveyard shift break 1","1:00 AM","1:15 AM","5200"
"Graveyard shift meal","3:00 AM","3:30 AM","5300"
"Graveyard shift break 2","5:00 AM","5:15 AM","5400"
"Day shift break 1","9:00 AM","9:15 AM","6200"
"Day shift meal","11:00 AM","11:30 AM","6300"
"Day shift break 2","1:00 PM","1:15 PM","6400"
"Swing shift break 1","5:00 PM","5:15 PM","7200"
"Swing shift meal","7:00 PM","7:30 PM","7300"
"Swing shift break 2","9:00 PM","9:15 PM","7400"
```

© Inductive Automation

2.3.4 Downtime Reasons

Downtime reasons allow the tracking of specific causes preventing a *line* or cell from running. Some reasons are considered causes of downtime where others are not. For example, if the production cell outfeed is backed up and there is no room to discharge product to, then it must shutdown. In this example, it is simply normal operation for the cell and it is not causing the production *line* from producing product. A cell further down the *line* is the cell preventing the production *line* from producing product.

Other downtime reasons may be planned. Any time that the production *line* is scheduled around, such as breaks, lunches, safety meeting, disable shifts, etc., is planned and will not count against the production *line* OEE Availability.

The OEE Downtime and Scheduling module has been designed to accommodate a variety of methods to determine reasons that a production *line* is down. This was done because monitoring all downtime reasons automatically is the ideal solution. But in the real world, this be difficult, costly, or just not practical to detect downtime reasons automatically. For this reason it is important for downtime tracking software to support both automatic reason detection and a manual override. For example: if an operator presses the stop button because they see a bottle laying on its side feeding into a filler, then the only automatic reason that can be detected is "operator pressed stop button". Now, the operator should be able to override the reason with more specific information.

In applications where the production cell is not automated and work is performed completely by manual labor, all downtime information can be entered manually from a predetermined list.

Downtime Reason Detection

For this reason, the OEE Downtime and Scheduling module determines the downtime reason from a single numeric value. Single numeric values are stable and can only represent one state. Of course one could use Expressions or script in Ignition to evaluate multiple values from the PLC and calculate a single numeric value representing the downtime reason, but this degrades the reliability of determining downtime reasons. Another benefit is that it is typically faster and reduces network traffic to read one value as opposed to multiple scattered values from a PLC.

The reason code with the numeric value of 0 is reserved for idle and 1 is reserved to mean running. All other reason codes are available for downtime reasons and is only limited by the maximum numeric value your PLC can handle. When the OEE Downtime and Scheduling module detects a production *line* or cell state that changed from a value of 1 (running), it will lookup the downtime reason from the state value. If communication to the PLC fails, in the case when a electrical disconnect is shut off, the production *line* or cell state is replaced with 0. If this happens during a production run, it will count as downtime.

Important:

Some systems may accommodate boolean logic to determine the downtime cause. However, consider the various values from a PLC that are going to be used to determine the downtime reason. These scattered values may come in from the PLC at different times and if the boolean logic resided in the OEE Downtime and Scheduling module, then it may be determining the reason on partially current values. Oops, now we have the incorrect reason and when all of the current values do arrive, what do we do? Do we change the original reason, add a new downtime entry, or maybe put a delay in to allow for all of the current values to arrive? None of these options are good solutions.

Automatic Detection

When the value of the State SQLTag changes to a value that is other than the numeric value of one, the system will look for a matching reason code in the entries in downtime reasons table. If it is not found it

© Inductive Automation

will replace then reason code with zero (0).

Manual Override

After an automatic reason has been triggered, the operator can override it will a more specific reason. Both are logged and can be viewed in analysis and reporting. For details about how to disable manual override see the *Editable* property in the Down Time Table section

Manual Only

For production *line*s that do not support automatic downtime detection, a completely manual implementation can be setup. This is done by providing a *line* drop-down list, or other component, on the operator screen that the user can use to select the current *line* state.

Line Downtime Versus Cell Downtime

It is important to understand the difference between *line* downtime and cell downtime. *Line* downtime, which is only the downtime reasons that are preventing the production *line* from producing product, is typically used to zero in and improve OEE. The cell downtime is used to look at trends and detect maintenance issues before they cause *line* downtime. Consider a production *line* that has 25 cells. If 5 of the cells are down all at the same time for unrelated reasons and only one of them is preventing product from being produced on the *line*, then there will be a lot of noise (extra irrelevant data) to weed through. Also, if a faster downstream cell stops, restarts and catches up, it may never affect the production of the *line* as a whole. The OEE Downtime Module provides the best of both worlds and tracks both *line* downtime and cell downtime.

For settings controlling cell downtime, see Cell Configuration under the Cell Downtime Settings section.

Short Downtime versus Long Downtime

Short downtime are events that last a small specified time, 120 seconds for instance. Short events will not affect the OEE availability calculation. The OEE Downtime module provides this threshold on a per line basis. If set to 0 then all downtime events are considered long and will always affect the OEE availability calculation.

Detecting Line Downtime

In the OEE Downtime Module, there are multiple options for detecting *line* downtime reasons. The options have been added to accommodate the wide variety of manufacturing processes. Below is a description of each method along with the situations where it can be used.

As you read through the methods described below, think of the effort required to manually implement them, whether done in the PLC or in Ignition.

Initial Reason

The concept of this method is the first cell that went down for a unplanned reason is the cause for the *line* not being able to produce product.

When a cell first goes down, the date and time is recorded. If multiple cell are down, they will each have their own date and time that it went down. The data and time for each down cell is looked at to determine the initial cell that went down and will be assigned as the cell causing the *line* downtime along with its reason. If the initial cell restarts, then the other down cells are looked the next cell in chronological order that went down. If there are two or more cells that went down at the same time, then it will use the order that they appear in the designer.

If there are no cells down for an unplanned reason, then the *line* will return to running state.

© Inductive Automation

This method should be used if all cells interact with one another. If any cell is down, then all other cells have to stop. A continuous liquid mixing process where at each cell, new ingredients are added or mixing or some other action is being performed fits into this category. If one cell stops, then all other upstream cells have to stop because there is no where to put the liquid and all downstream cells have to stop because there is not liquid to process. In this case the first cell that stopped is the cause for all other cells to stop.

Key Reason

This method uses the flow of the *line* to determine the cause for the *line* not being able to produce product. It also assumes there is a primary cell that, if down, will cause the *line* to stop producing product. This method also uses the order of the cells as they are configured in the designer. If the first cell is down for a reason that is not configured as *Record Downtime*, the next cell will be checked. If it is down for a reason that is configured as *Record Downtime*, then it will be assigned as *line* downtime cell and reason. When the second cell that caused the *line* downtime restarts but the first cell has not started yet because its discharge is still backed up, then the original cell and reason will still be the cause of downtime until the first cell restarts.

The concept behind this is that a faster downstream cell can go down, restart and catch up without ever causing loss of production on the *line*.

This method should be used for packaging *lines*. If the first cell on the *line* keeps accepting raw material, then the *line* will be producing product. However, in some situations, it could be the slowest machine because it cannot catch up for lost production.

Line State

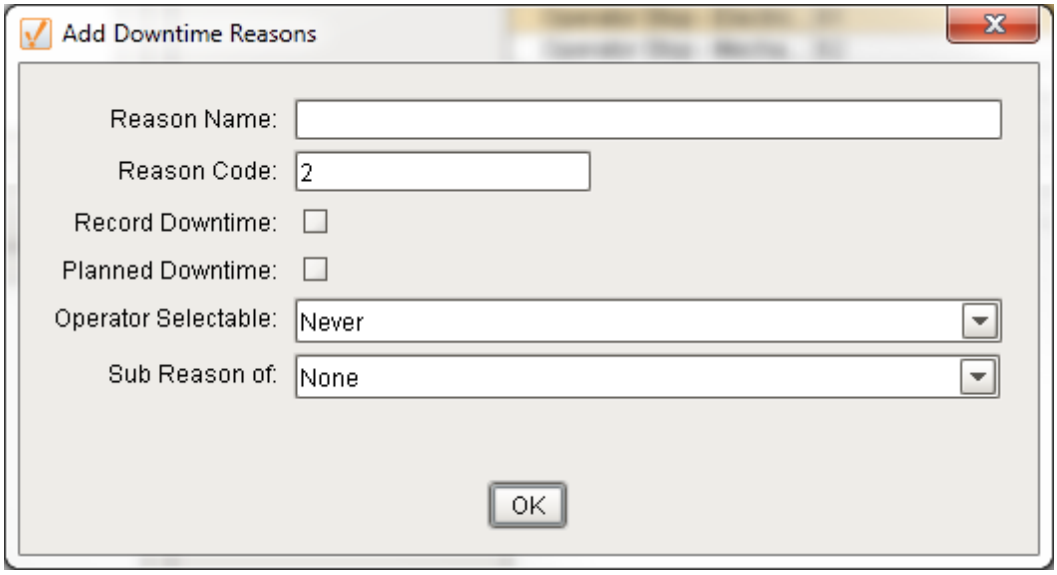
This method is used when the other methods are not appropriate. This method allows implementing custom methods of *line* downtime detection. When using this method, all downtime reasons must be entered into the *line* downtime reason table and not the cell downtime reasons table. This method will only read the *line* downtime reason from the State SQLTag configured for the *line* to determine the *line* downtime reason.

When using this method, detailed cell downtime tracking can still be used but it is isolated from the *line* downtime reasons.

2.3.4.1 Adding a Downtime Reasons

To add a Downtime Reason, right-click anywhere in the Downtime Reasons table, and select "New" from the menu. The following window will appear:

© Inductive Automation



Adding a Downtime Reason

Reason Name

The required reason name is used to reference one reason from another an must be unique within the production *line* or cell. The reason name should be meaningful to the end user. This is because the end user can filter and group analysis and report by the reason name.

Reason Code

The reason code is a required unique number to the cell that identifies the downtime reason. PLCs and other equipment are more apt to handing numbers versus strings, therefore a reason code is used for reference within the program.

The reason code 0 is reserved for idle.
The reason code 1 is reserved for running.

Record Downtime

If the ***Record Downtime*** option is true, then downtime events with this reason will be treated as unplanned downtime. This allows for downtime reasons such as *outfeed backup* to not be counted as unplanned downtime.

Planned Downtime

This option will make the reason Planned Downtime, meaning it is scheduled and will not be used in computing the OEE.

Operator Selectable

This option selects if or how this reason may be selected by an operator. The options are:

Never

This reason will not be available to a user to over-ride any other reason. It will display only if it is the original reason code detected by the state tag.

Always

This reason can be selected by a user to over-ride the original reason code detected.

Cannot Be Overridden

© Inductive Automation

When this is the originally detected reason code it cannot be over-ridden with any other code.

If Parent Reason Detected

This reason will be available for user selection only if the originally detected reason is the parent of this sub-reason (see Sub Reason of below).

Sub Reason Of

This option allows the user to to enter a new downtime reason as a child of another downtime reason. The reasons are sorted by the Reason Code in numerical order, but child reasons will always be sorted under their parent reason.

Reason Name	Reason Code	Record Downtime	Planned Downtime	Operator Selectable
Stop	0	true	false	Never
Machine Fault	3	true	false	Never
Machine Fault - Electric...	31	true	false	If Parent Reason Detected
Machine Fault - Mecha...	32	true	false	If Parent Reason Detected
Outfeed Backup	4	false	false	Never
Waiting For Product	6	true	false	Never
Scale Fault	8	true	false	Cannot Be Overridden
Over Temperature	9	true	false	Never
Scale Maintenance	20	true	false	Always

Example of a Sub Reason

2.3.4.2 Editing a Downtime Reasons

To edit a Downtime Reason, select the existing Downtime Reason you wish to edit, then right-click and select "Edit" from the menu. The same window used to add downtime reasons will appear, allowing the information to be edited.

2.3.4.3 Deleting a Downtime Reasons

To delete a Downtime Reason, select the existing Downtime Reason you wish to remove, then right-click and select "Delete" from the menu. A window will appear confirming that you permanently want to delete the downtime reason.

2.3.4.4 Import / Export

To import downtime entries, right-click anywhere on the downtime table and select the **Import** menu item. A dialog box will appear to allow selection of a comma separated values (csv) formatted file.

The first line of the file must at least contain the property names separated by commas. If additional names exist, they will be ignored. The property names can be in any order. Below is a sample csv file showing multiple downtime entries.

```
Reason Name,Reason Code,Record Downtime,Planned Downtime,Operator
Selectable,Sub Reason Of
"Stop","0","true","false","0","-1"
"Machine Fault","3","true","false","0","-1"
"Machine Fault - Electrical","31","true","false","1","3"
"Machine Fault - Mechanical","32","true","false","1","3"
"Outfeed Backup","4","false","false","0","-1"
"Waiting For Product","6","true","false","0","-1"
```

© Inductive Automation

```
"Scale Fault","8","true","false","3","-1"
"Over Temperature","9","true","false","0","-1"
"Scale Maintenance","20","true","false","2","-1"
"Container Jam","22","true","false","2","-1"
"Planned Shutdown","99","false","true","2","-1"
"Meal","100","false","true","2","-1"
"Break","101","false","true","2","-1"
```

To export downtime entries, right-click anywhere on the table containing downtime entries and select the **Export** menu item. A dialog box will appear to allow selection of an existing file or the typing in of a name of the new file to save the downtime entries to. If a file extension is not entered, then the default .csv will be used.

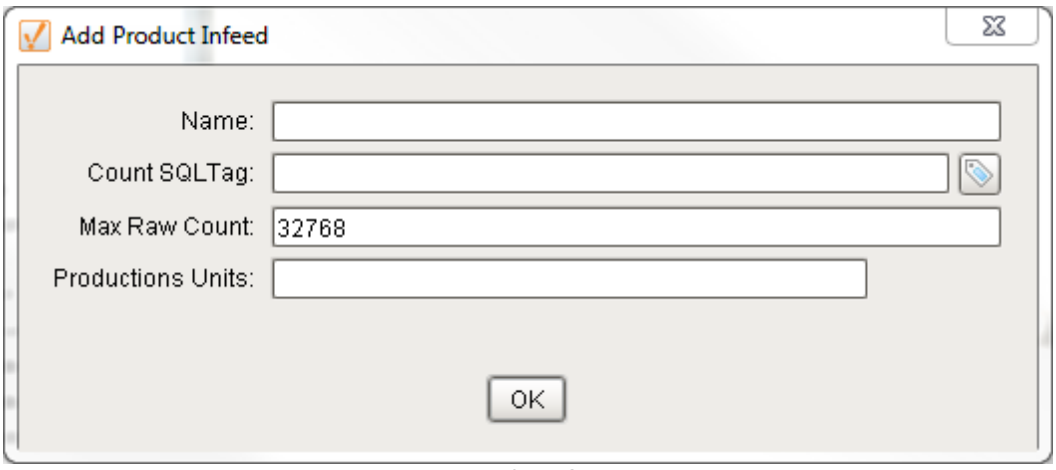
2.3.5 Product Infeed

Product infeeds are used only to calculate waste or if infeed rate information is desired. This applies to both production *lines* and production cells. If a production *line* or cell is configured for more than one infeed or outfeed, then accumulation and waste calculations will be performed for each combination. For example, a production *line* can be configured to track containers, caps and product as infeeds and a single outfeed of full containers. The independent waste calculations for containers, caps and production will be performed. See Production Count Tracking section for more information.

For each infeed, the OEE module will start calculating production rate per minute, or production rate per hour values. These values can be accessed through the Production OPC Server. See the section on Production OPC Values for more information.

2.3.5.1 Adding a Product Infeed

To add a product infeed entry, right-click anywhere on the product infeed table of a production *line* or cell and select the **New** menu item. A dialog box will appear to allow entry of a name, count sql tag, maximum raw count and production units as shown below.



Product Infeed Settings

Name

The required infeed name is used to reference one infeed from another and must be unique.

Count SQLTag

The required SQLTag is the source of the raw production counts. This typically comes from a PLC, but can come from other sources such as barcode readers, database queries or derived by another means. The data type (format) of the SQLTag containing the raw production count must be a number.

Max Raw Count

This is the maximum raw count value before it is reset to zero. See note below.

Production Units

This can be anything you want that represents the units. Examples are: gallons, cases, bottles, pounds, liters, etc.

Note: The term raw count is used because it is a relative production count. It just starts at zero and counts up to a rollover value, typically 32767, where it becomes zero again. The OEE Downtime and Scheduling module calculates the actual production count from raw count. This eliminates having to reset the value in the PLC, or other device, at the beginning of a production run. As a result, the programming that is required in the PLC, or other device is simplified. It also eliminates problems typically associated with reset handshaking and production runs that exceed the limits of PLC counters. For an OEE tracking system to be accurate, it must withstand communication errors power outages, etc. By using raw counts that rollover and let the OEE Downtime and Scheduling module handle the actual production count, the system is robust. Besides, that is just less PLC programming that has to be done and tested.

2.3.5.2 Editing a Product Infeed

To edit an existing product infeed entry, right-click on the desired entry in the product infeed table of a product *line* or cell and select the **Edit** menu item. A dialog box similar to the add dialog box will appear, allowing editing of the entry.

2.3.5.3 Deleting a Product Infeed

To remove an existing product infeed entry, right-click on the desired entry in the product infeed table of a production *line* or cell and select the **Delete** menu item. A window will appear confirming that you want to remove the product infeed entry.

2.3.5.4 Import / Export

To import product infeed entries, right-click anywhere on the product infeed table and select the **Import** menu item. A dialog box will appear to allow selection of a comma separated values (csv) formatted file.

The first line of the file must at least contain the property names separated by commas. If additional names exist, they will be ignored. The property names can be in any order. Below is a sample csv file showing a single product infeed entry.

```
Name, Count SQLTag, Max Raw Count, Productions Units
Line Infeed, Line 1/PLC/Filler/Infeed, 32768, Bottles
```

To export product infeed entries, right-click anywhere on the table containing product infeeds and select the **Export** menu item. A dialog box will appear to allow selection of an existing file or the typing in of a name of the new file to save the product infeed entries to. If a file extension is not entered, then the default .csv will be used.

© Inductive Automation

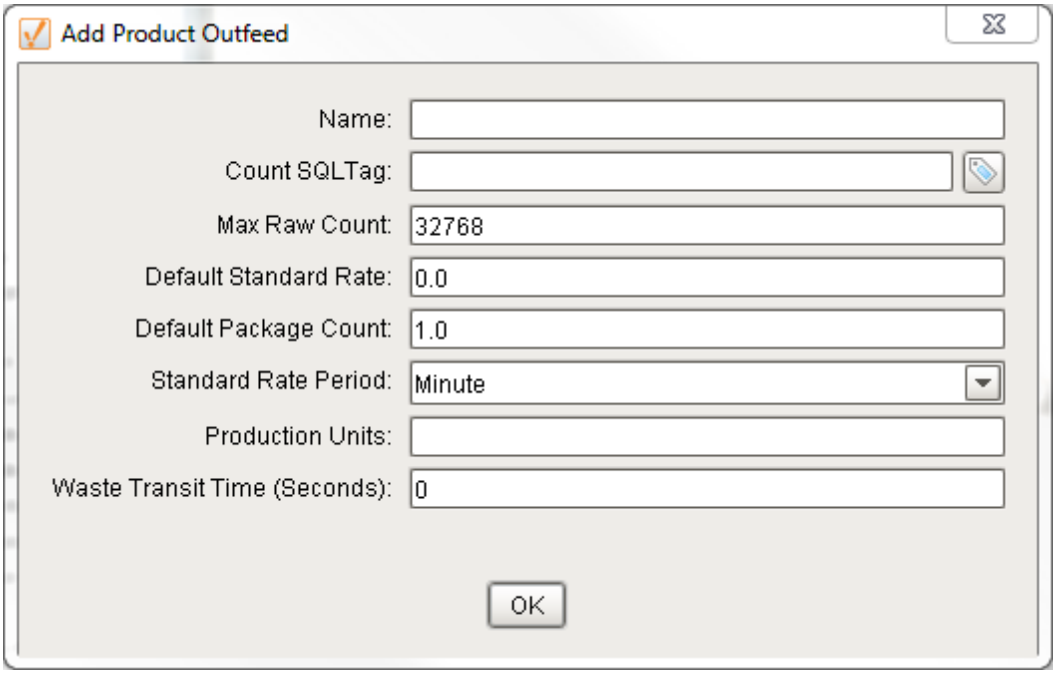
2.3.6 Product Outfeed

Before OEE calculations can be performed, production count information is required. At a minimum, the outfeed production count for a production *line* is needed. Additional production count information can be configured that will result in more OEE calculations. For example, if the infeed production count is configured for a production, then product accumulation and waste is calculated. See Production Count Tracking section for more information.

For each outfeed, the OEE module will start calculating production rates, OEE, etc. values. These values can be accessed through the Production OPC Server. See the section on Production OPC Values for more information.

2.3.6.1 Adding a Product Outfeed

For each outfeed, the OEE module will start calculating the production rate per minute, or production rate per hour values. These values can be accessed through the Production OPC Server. See the section on Production OPC Values for more information. To add a product outfeed entry, right-click anywhere on the product outfeed table of a production *line* or cell and select the **New** menu item. A dialog box will appear to allow entry of the new information, as shown below.



Product Outfeed Settings

Name
The required outfeed name is used to reference one outfeed from another and must be unique.

Count SQLTag
The required SQLTag is the source of the raw production counts. This typically comes from a PLC, but can come from other sources such as barcode readers, database queries or derived by another means. The data type (format) of the SQLTag containing the raw production count must be a number.

Max Raw Count
This is the maximum raw count value before it is reset to zero. See note below.

© Inductive Automation

Default Standard Rate

The OEE calculation requires the designed rate that the production *line* can produce. Typically, machines and processes only run at these rates theoretically. This setting is the default value for the standard rate but can be overridden by product and *line* in the user screens.

Default Package Count

This is the default number of infeed units which end up in an outfeed unit. If package count does apply, then enter 1.0. For example, there may be 10 bottle (infeed) in a case (outfeed) or 10 gallons (infeed) in a bucket (outfeed).

When calculating waste and production count information, the package size is very important. It can change based on the product being run and the default value, and can be overridden by the product in the user screens.

Standard Rate Period

This is the time period to use for the default standard rate If the default standard rate is in units per hour, select **Hour** otherwise select **Minute.**

Production Units

This can be anything you want that represents the units. Examples are: gallons, cases, bottles, pounds, liters, etc.

Waste Transit Time (Seconds)

The waste transit time specifies the amount of time it takes for one unit to travel from the infeed to the outfeed if the production *line* is running at standard rate. It is used to calculate the waste count.

Note: The term raw count is used because it is a relative production count. It just starts at zero and counts up to a rollover value, typically 32767, where it become zero again. The OEE Downtime and Scheduling module calculates the actual production count from raw count. This eliminates having to reset the value in the PLC, or other device, at the beginning of a production run. As a result, the programming that is required in the PLC, or other device is simplified. It also eliminates problems typically associated with reset handshaking and production runs that exceed the limits of PLC counters. For an OEE tracking system to be accurate, it must withstand communication errors power outages, etc. By using raw counts that rollover and let the OEE Downtime and Scheduling module handle the actual production count, the system is robust. Besides, that is just less PLC programming that has to be done and tested.

2.3.6.2 Editing a Product Outfeed

To edit an existing product outfeed entry, right-click on the desired entry in the product outfeed table of a product *line* or cell and select the **Edit** menu item. A dialog box similar to the add dialog box will appear allowing editing of the entry.

2.3.6.3 Deleting a Product Outfeed

To remove an existing product outfeed entry, right-click on the desired entry in the product outfeed table of a production *line* or cell and select the **Delete** menu item. A window will appear confirming that you want to remove the product outfeed entry.

2.3.6.4 Import / Export

To import product outfeed entries, right-click anywhere on the product outfeed table and select the **Import** menu item. A dialog box will appear to allow selection of a comma separated values (csv) formatted file.

© Inductive Automation

The first line of the file must at least contain the property names separated by commas. If additional names exist, they will be ignored. The property names can be in any order. Below is a sample csv file showing a single product infeed entry.

```
Name, Count SQLTag, Max Raw Count, Default Standard Rate, Default Package Count, Standard Rate Period, Production Units, Waste Transit Time (Seconds)
Line Outfeed, Line 1/PLC/Palletizer/Outfeed, 32768, 60.0, 10.0, Hour, Cases, 120
```

To export product outfeed entries, right-click anywhere on the table containing product outfeeds and select the **Export** menu item. A dialog box will appear to allow selection of an existing file or the typing in of a name of the new file to save the product outfeed entries to. If a file extension is not entered, then the default .csv will be used.

2.3.7 Product Waste

Before OEE Quality calculations can be performed, waste count information is required. Because of the varied approaches of determining waste on a production *line* or process, the OEE Downtime Module allows different methods of collecting waste information.

1. Use the Run Waste Count OPC value for the *line*. With this method, the OEE Downtime Module will simply use the current value of the Run Waste Count OPC value when calculating the OEE Quality values. This provides for custom waste tracking or calculations if the methods built in to the OEE Downtime Module don't fit your requirements. If OEE Quality is not being used, then this method should be used and setting the Run Waste Count OPC value to zero.

2. Automatically calculate the waste count using the built-in algorithm based on the infeed count, outfeed count and transit time defined in the Product Outfeed. This method is an approximation and is less accurate especially in cases when product accumulation sections are used on the *line*.

3. Use configured Product Waste counters. This OEE Downtime Module will track waste count using the same method used for infeed and outfeed counts. The waste counts will be totalized and used in the OEE Quality calculations.

Waste Calculation Methods

There are several ways to calculate the amount of waste coming from a production line:

None
No waste calculation methods will be used to determine waste counts.

Run Waste Count Tag
Waste will be calculated based on what is entered in the Run Waste Count tag. This may be a unique formula used to calculate waste or a manual entry of the waste count.

Started vs. Finished
When this method is used, outfeed will be compared to infeed to determine how many units were lost to waste. If 1000 units enter the production line, and 900 finished products exit the production line, then it is assumed that 100 units were lost to waste. A Waste Transit time other than 0 must be entered for the primary outfeed. The Waste Transit time should be the normal amount of time a production unit takes to travel from infeed (started) to outfeed (finished).

Product Waste Entries

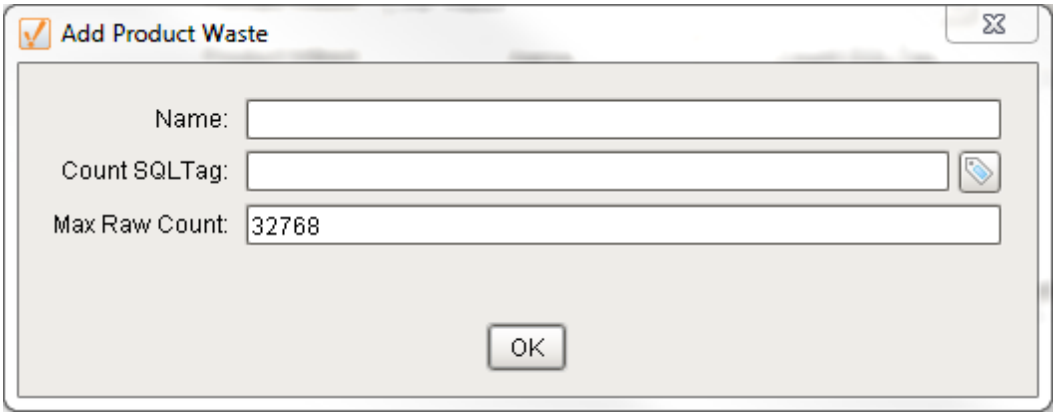
© Inductive Automation

In order to calculate waste by this method, one or more entries must be in the Product Waste table. The waste from each entry is added together to equal the total amount of waste. For example, there may be two cells on the line that inspect and discard faulty products. If the first cell discarded 10 units, and the second cell discarded 5 units, the total waste count would be 15 units.

Sum Product Waste From Each Cell
This method will sum up each of the cell's waste values. Product Waste Entries will be ignored.

2.3.7.1 Adding a Product Waste Counter

To add a product waste entry, right-click anywhere on the product waste table of a production *line* or cell and select the **New** menu item. A dialog box will appear to allow entry of a name, count SQLTag and maximum as shown below.



Product Waste Settings

Name
The required product waste name is used to reference one waste entry from another and must be unique.

Count SQLTag
The required SQLTag is the source of the raw waste counts. This typically comes from a PLC, but can come from other sources such as barcode readers, database queries or derived by another means. The data type (format) of the SQLTag containing the raw waste count must be a number.

Max Raw Count
This is the maximum raw count value before it is reset to zero. See note below.

Note: The term raw count is used because it is a relative waste count. It just starts at zero and counts up to a rollover value, typically 32767, where it become zero again. The OEE Downtime and Scheduling module calculates the actual waste count from raw count. This eliminates having to reset the value in the PLC, or other device, at the beginning of a production run. As a result, the programming that is required in the PLC, or other device is simplified. It also eliminates problems typically associated with reset handshaking and production runs that exceed the limits of PLC counters. For an OEE tracking system to be accurate, it must withstand communication errors power outages, etc. By using raw counts that rollover and let the OEE Downtime and Scheduling module handle the actual waste count, the system is robust. Besides, that is just less PLC programming that has to be done and tested.

2.3.7.2 Editing a Product Waste Counter

To edit an existing product waste entry, right-click on the desired entry in the product waste table of a product *line* or cell and select the **Edit** menu item. A dialog box similar to the add dialog box will appear allowing editing of the entry.

© Inductive Automation

2.3.7.3 Deleting a Product Waste Counter

To remove an existing product waste entry, right-click on the desired entry in the product waste table of a production *line* or cell and select the **Delete** menu item. A window will appear confirming that you want to remove the product waste entry.

2.3.7.4 Import / Export

To import product waste entries, right-click anywhere on the product waste table and select the **Import** menu item. A dialog box will appear to allow selection of a comma separated values (csv) formatted file.

The first *line* of the file must at least contain the property names separated by commas. If additional names exist, they will be ignored. The property names can be in any order. Below is a sample csv file showing a single product waste entry.

```
Name, Count SQLTag, Max Raw Count
Rejector, Line 1/PLC/RejectorCount, 32768
```

To export product waste entries, right-click anywhere on the table containing product waste entries and select the **Export** menu item. A dialog box will appear to allow selection of an existing file or the typing in of a name of the new file to save the product waste entries to. If a file extension is not entered, then the default .csv will be used.

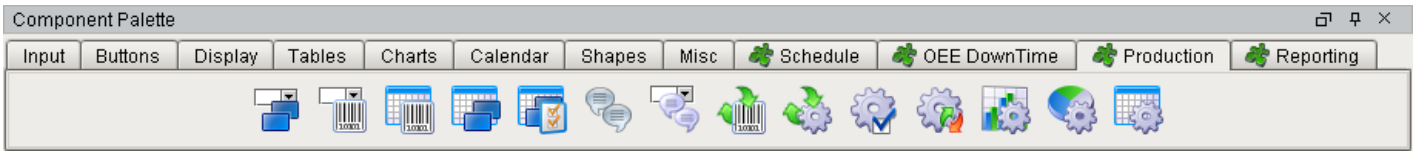
2.4 Component Reference

This section will describe the components that are available with the OEE Downtime and Scheduling module.

Please note that only the properties, methods and events that are specific to the OEE Downtime and scheduling module components are described here. For description and usage of other properties see the Ignition reference manual.

2.4.1 Production Components

When the Production Module, which is part of the OEE Downtime and Scheduling Module, is opened, a new component tab will appear. On it are a number of components that provide functionality specific to the production model, product codes, analysis, etc.

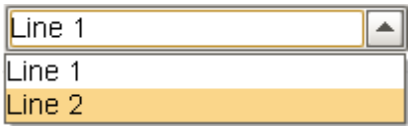


Production Components

2.4.1.1 Production Line Selector

Description

A component that provides users to select a production *line* from a drop-down list. Production *lines* are defined in the production model within the designer.



Line Drop-Down List

Properties

This component has standard Ignition properties with the addition of the following properties:

Selected Line Path

The currently selected *line* path. This is the full path name of the *line* starting with the project name.

For example: "OEEDemo\Your Enterprise\Your Site\Your Area\Line 1"

```
Scriptiselected LinePath
ng name
Data    String
Type
```

Selected Line Name

The currently selected *line* name. This is just the *line* name excluding the rest of the *line* path.

For example: "Line 1"

```
Scripting name     selectedLineName
Data Type          String
```

The currently selected *line* ID. This is internal system ID for this line. Useful for user defined queries into the database.

For example: "event.source.parent.getComponent('Production Line Selector').selectedLineID" will return the selected line id.

```
Scripting name     selectedLineID
Data Type          String
```

Line Filter

A comma separated list of lines to display.

Example: When set to "Line A, Line B" then only Line A and Line B will be displayed.

```
Scripting name     lineFilter
Data Type          String
```

Production Model Item Filter

The production model path to filter upon.

Example: When set to "OEEDemo\Your Enterprise\Site 1\Packaging" then all lines under Packaging will be displayed.

```
Scripting name     itemPath
Data Type          String
```

Events

This component has standard Ignition events.

© Inductive Automation

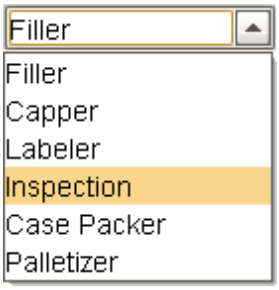
Methods

(none)

2.4.1.2 Production Cell Selector

Description

A component that provides users to select a production cell from a drop-down list. Production cells are defined in the production model within the designer.



Cell Drop-Down List

Properties

This component has standard Ignition properties with the addition of the following properties:

Line Path

The *line* path of the production *line* that this component is associated with. This is the full path name of the *line* starting with the project name. Only the cells for this *line* path will be shown in this component.

For example: "OEEDemo\Your Enterprise\Your Site\Your Area\Line 1"

```
Scripting name          linePath
Data Type               String
```

Selected Cell Path

The currently selected cell path. This is the full path name of the cell starting with the project name.

For example: "OEEDemo\Your Enterprise\Your Site\Your Area\Line 1\Filler"

```
Scripting name          selectedCellPath
Data Type               String
```

Selected Cell Name

The currently selected cell name. This is just the cell name excluding the rest of the cell path.

For example: "Filler"

```
Scripting name          selectedCellName
Data Type               String
```

Events

This component has standard Ignition events.

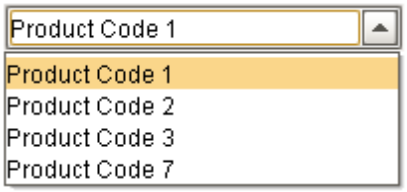
Methods

(none)

2.4.1.3 Product Code Selector

Description

A component that provides users to select a product code from a drop-down list of available product code for a production *line*. Product code information is stored in the "ProductCode", "ProductCodeLine", "ProductCodeLineProperty" database tables. The Product Code Table, Product Code Line Table and Product Code Properties Table are typically used to manage the information in these database tables eliminating the need for SQL statements and scripts to do so.



Product Code Drop-Down List

Properties

This component has standard Ignition properties with the addition of the following properties:

Line Path

The *line* path of the production *line* that this component is associated with. This is the full path name of the *line* starting with the project name.

For example: "OEEDemo\Your Enterprise\Your Site\Your Area\Line 1"

```
Scripting name          linePath
Data Type               String
```

Selected Product Code

The currently selected product code ID.

```
Scripting name          selectedStringValue
Data Type               String
```

Events

This component has standard Ignition events.

Methods

(none)

2.4.1.4 Product Code Table

Description

A component that displays all the available product codes in a table and allows the product code to be disabled. All product codes are automatically displayed from the "ProductCode" database table without the need for custom SQL statements or script.

Product Code	Description	Disable
Product Code 1		☐
Product Code 2		☐
Product Code 3		☑
Product Code 7		☐

Product Code Table

When a product code is disabled then it cannot be selected during work order creation or product code selection.

This component usually works in conjunction with the Product Code Line Table and Product Code Properties Table components. Refer to the OEEDemo project for a complete example.

© Inductive Automation

Properties

This component has standard Ignition properties with the addition of the following properties:

Selected Product Code The currently selected product code from the table.

```
Scripting name          selectedProductCode
Data Type               String
```

Selected Product Code ID The currently selected product code ID. This is the ID for the "ProductCode" database table.

```
Scripting name          selectedProductCodeID
Data Type               String
```

Hide Disabled Product Codes If set to True then disabled Product Codes will be hidden from the table.

```
Scripting name          hideDisabled
Data Type               Boolean
```

Product Code Filter Filters the results in the table that begin with the given string. If left blank all product codes are returned.

```
Scripting name          productCodeFilter
Data Type               String
```

Events

This component has standard Ignition events.

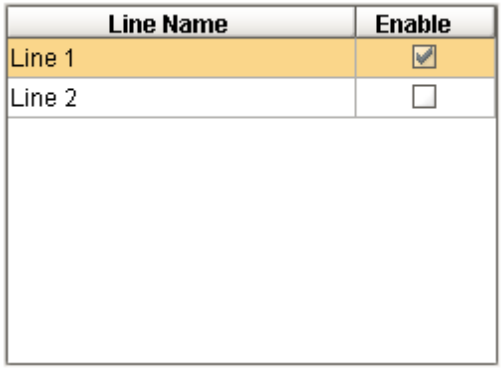
Methods

```
(none)
```

2.4.1.5 Product Code Line Table

Description

This component displays all the available *lines* and allows the linked product code to be enabled to be run on production *lines*. All product code lines are automatically displayed from the "ProductCodeLine" database table without the need for custom SQL statements or script.



When a *line* is enabled for a product code, it will show up it the list of available products when scheduling, etc. for that *line*.

This component usually works in conjunction with the Product Code Table and Product Code Properties Table components. Refer to the OEEDemo project for a complete example.

Properties

This component has standard Ignition properties with the addition of the following properties:

Product Code ID

The currently selected product code ID. This is the ID for the "ProductCode" database table. Normally this is bound to the Product Code Table "Selected Product Code ID".

```
Scripting name          productCodeID
Data Type               String
```

Selected Product-CodeLine ID

Value of the currently selected product code internal ID. This is the ID for the "ProductCodeLine" database table.

```
Scripting name          selectedProductCodeLineID
Data Type               String
```

Selected Line Name

Value of the currently selected *line* name.

```
Scripting name          selectedLineName
Data Type               String
```

Events

This component has standard Ignition events.

Methods

```
(none)
```

2.4.1.6 Product Code Properties Table

Description

This component displays, and allows editing of, property values for specific product code and production *line* combination. This is where standard rates and scheduling rates are defined by product code and production *line*.

The properties that appear depend on the production model configuration done in the designer. There will be properties for the production *line* at the top followed by properties for each production cell.

© Inductive Automation

Property	Value	Default Value
Line 1		
Line Outfeed.Package Count	1.0	1.0
Line Outfeed.Standard Rate	3500	3600.0
Schedule Rate	3400	3600.0
Filler		
CellEnabled	☑	TRUE
Outfeed.Package Count	1.0	1.0
Outfeed.Standard Rate	3600.0	3600.0
Capper		

Product Code Properties Table

The Value column will indicate the property setting value and allow editing the of value for the specified *line*. The default value is for reference and is not editable. The values are saved in the "ProductCodeLineProperty" database table.

This component usually works in conjunction with the Product Code Table and Product Code Line Table components. Refer to the OEEDemo project for a complete example.

Properties

This component has standard Ignition properties with the addition of the following properties:

Product Code Line ID

The product *line* ID. This is the ID for the "ProductCodeLine" database table. Normally this is bound to the Product Code Line Table "Selected Product Code Line ID".

```
Scripting name          productCodeLineID
Data Type               String
```

Events

This component has standard Ignition events.

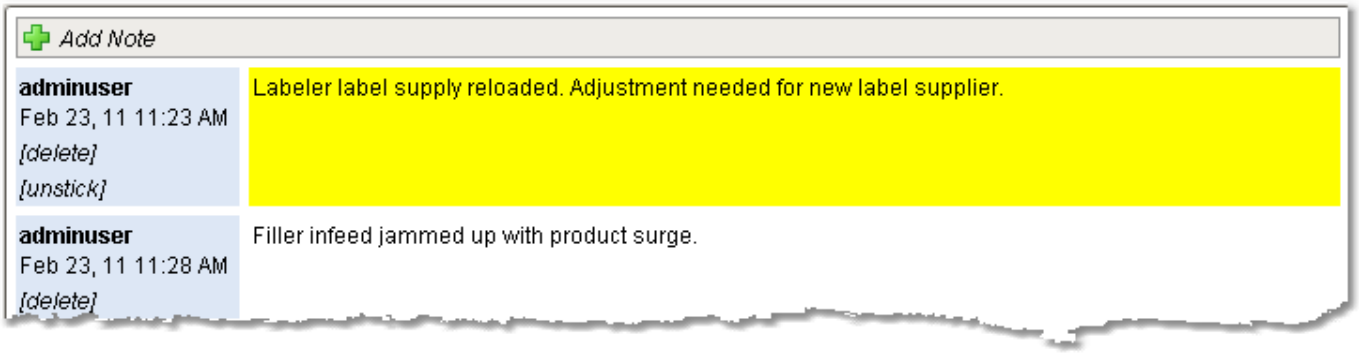
Methods

```
(none)
```

2.4.1.7 Production Comments Panel

Description

A component that allows comments/notes to be entered for the current production run. This component is similar to the Ignition Comments Panel component, but eliminates the need for SQL statements or scripting. It ties comments to the production run that the production *line* is currently running.



Production Comments Panel

To add a comment select the "+ Add Note" link. A new window panel will appear and allow you to enter text.

If you select "Sticky?" that will force the note(s) to appear at the top of the list. The color of the background of a sticky note can be controlled with the "Sticky Note Color" property.

After a sticky note is entered, it can be "un-stuck" by selecting the "*[unstick]*".

If note deletion is allowed, the link "*[delete]*" can be selected to delete the note.

Properties

This component has standard Ignition properties with the addition of the following properties:

Line Path

The *line* path of the production *line* that this component is associated with. This is the full path name of the *line* starting with the project name.

For example: "OEEDemo\Your Enterprise\Your Site\Your Area\Line 1"

```
Scripting name          linePath
Data Type               String
```

Run Reference ID

The run ID of the production run to display comments for. If its value is set to -1, then comments for the current production run will be displayed.

Note: Setting of this property is only required when viewing comments for past production runs.

```
Scripting name          refID
Data Type               int
```

Delete Mode

Determines how deleting of comments will be handled.

```
Scripting name          deleteMode
Data Type               int
Values                  No Deletes
                        Owner Deletes
                        Any Deletes
```

© Inductive Automation

Entered By

Allows a custom value to be used to indicate who entered the comment. If left blank it will use the currently logged in user name.

Scripting name	enteredByName
Data Type	String

Events
This component has standard Ignition events.

Methods

`newNote()`

New note.

parameters	(none)
returns	nothing

2.4.1.8 Product Code Controller

Description
An invisible component that provides adding product codes. The term *invisible component* means that the control appears during design time, but is not visible during runtime. Product codes are stored in the "ProductCode" database table and this control handles all SQL statements, duplicate checking, etc.

Alternatively, product codes can added directly into the "ProductCode" database table directly, bypassing the OEE Downtime and Scheduling Module. This method supports integration to ERP or other software systems.

Product codes can also be added via scripting.

Properties
This component has standard Ignition properties.

Events
This component has standard Ignition events.

Methods

`addProductCode(productCode, description)`

Add new production code and description to database.

parameters

productCode	The product code to add to the database
	
	`Data Type`
	`String`

description	A descriptive label for the product code
	
	`Data Type`
	`String`

returns

message	contains a description of any error encountered, usually that the

product code already exists. Otherwise it will be empty.

Data Type
String

Example Code

The following script can be entered in a button's actionPerformed event. It will add the product code and description to the database. The return message will indicate if the there are any issues adding the product code, such as if the product code already exists.

```
message = event.source.parent.getComponent('Product Code Controller')
.addProductCode(event.source.parent.getComponent('ProductCode').text,
event.source.parent.getComponent('ProductCodeDescription').text)
if message == '':
   system.nav.closeParentWindow(event)
else
  system.gui.errorBox(message)
```

2.4.1.9 Analysis Controller

Description

An invisible component that makes analysis data available for reports and other components. The term *invisible component* means that the control appears during design time, but is not visible during runtime.

In cases where the Production Analysis Selector offers too many options to the use, this component can be used. It has all of the same functionality as the Production Analysis Selector but without the user interface. This means property bindings or script must be used to make the filter, compare by and data point selections. It also is used for providing data to canned reports and optionally allowing the user to make limited filter options.

This component also contains methods for deleting and restoring a run.

Properties

This component has standard Ignition properties with the addition of the following properties:

Automatic Update

When true, when any property that changes the results will change, the results will automatically be updated.

Scripting name automaticUpdate
Data Type Boolean

Table Data

This property holds data in a format that is optimized for binding to a table component.

Scripting name tableData
Data Type Dataset

Data Format

This property specifies the type of data to return from the server.

Options:
Table - Only data optimized for tables will be included in the results.
Chart - Only data optimized for charts will be included in the results.
Both - Table and chart data will be included in the results.

```
Scripting name          dataFormat
Data Type               AnalysisDataFormat
values                  Table
                        Chart
                        Both
```

Chart Data

This property holds data in a format that is optimized for binding to pie and bar chart component such as the Production Bar Chart and Production Pie Chart.

```
Scripting name          chartData
Data Type               Dataset
```

Line Chart Data

This property holds data in a format that is optimized for binding to a *line* chart component.

```
Scripting name          lineChartData
Data Type               Dataset
```

Drill Down Options

This property holds the drill down options appropriate for the current filter and compare by settings.

```
Scripting name          drillDownOptions
Data Type               Dataset
```

Previous Drill Down Enabled

This property indicates if there are entries in the drill down cache maintained by this component.

```
Scripting name          previousDrillDownEnabled
Data Type               Boolean
```

Provider Name

This property holds the current provider of analysis data. See Analysis Providers for available options.

```
Scripting name          providerName
Data Type               String
```

Filter

This property holds the current filter item selections to filter the analysis results by. If more than one item exists, they are separated by commas. See Analysis Providers for available filters for each provider type.

```
Scripting name          filter
Data Type               String
```

© Inductive Automation

Compare By

This property holds the current compare by item selections to group the analysis results by. If more than one item exists, they are separated by commas. See Analysis Providers for available compare by values for each provider type.

```
Scripting name          compareBy
Data Type               String
```

Data Points

This property holds the currently selected data points to include in the results. If more than one item exists, they are separated by commas. See Analysis Providers for available data points for each provider type.

```
Scripting name          dataPoints
Data Type               String
```

Start Date

This property is the starting date for retrieving analysis data and determining available filter and compare by options.

```
Scripting name          startDate
Data Type               Date
```

End Date

This property is the ending date for retrieving analysis data and determining available filter and compare by options.

```
Scripting name          endDate
Data Type               Date
```

Dynamic Properties

Depending on the setting of the **Provider Name** property, the dynamic properties will change. A dynamic property to be created for each filter category that can be bound to by other components. These dynamic properties can also be set through script. See Analysis Providers for available filters for each provider type.

For example:
If the **Provider Name** property is set to "Downtime", then **Shift** will be created for one of the dynamic properties. The **Shift** dynamic property can be bound to a Dropdown List Component populated with 1, 2 and 3. Changing the selection of the drop-down list will change the analysis results to be filtered by the select shift.

Events

This component has standard Ignition events plus the following events:

progressStart

Is fired when this component begins to process property changes. Usefull for displaying an "in progress" message during processing.

Event Properties

event.source()

The component source of the event (this component).
Data Type Component

beforeUpdate

Event Properties

(none)

afterUpdate

Event Properties

progressEnd

Is fired when this component completes property changes. Usefull for displaying an "in progress" message during processing.

Event Properties

© Inductive Automation

© Inductive Automation

Methods

drillDown(drillDownName, item)

Sets all the analysis selections to new state dictated by the drill down definition.

parameters

drillDownName	A drill down definition name. This is typically supplied by th down event of one of the display components	
	Data Type	String
item	A drill down category. This is typically supplied by the drill of one of the display components	
	Data Type	Object

returns nothing

---

prevDrillDown()

Sets all the analysis selections to the previous state before the last drill down.

parameters (none)

returns nothing

---

update()

Causes the results to be updated.

parameters (none)

returns nothing

---

addDatasetColumn()

This method is used for reporting. Because the Ignition Report module does not support master slave table relationships, this method is used to add new columns containing a Dataset with child rows. For each row in the analysis controller results, a child Dataset will be created and placed into the new column named specified by the columnName parameter. The rows in the child Dataset are determined from the Dataset specified in the dataset parameter and match the column value specified by the keyColumns parameter.

parameters

dataset	Dataset containing child rows.	
	Data Type	Dataset
columnName	Name of column to add that will contain the child dataset.	
	Data Type	String
keyColumns	Name of columns to break the child row up by. Multiple ke can be specified by separating then with a comma.	
	Data Type	String

returns nothing

---

deleteRun(runUUID)

Marks the run as deleted for analysis purposes. This run and associated data will not appear in any analysis datasets.

© Inductive Automation

Must use "Run, Run Identifier" as the "Compare By" for this method to work.

parameters

run The unique Run Identifier for the run to delete.

Data Type

String

returns nothing

---

restoreRun(runUUID)

Restores a previously deleted run.
Must use "Run, Run Identifier" as the "Compare By" for this method to work.
Use "Deleted Runs=Show" as the filter to show runs that have been deleted.

parameters

run The unique Run Identifier for the run to delete.

Data Type

String

returns nothing

Example Code

This script would be entered into the "drillDown" event of a Production Bar Chart.

```
event.source.parent.getComponent('Production Analysis Selector')
.drillDown(event.getDrillDownName(), event.getCategory())
```

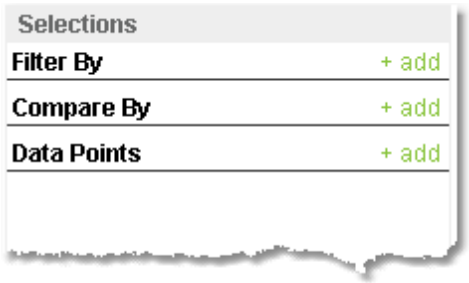
This script would be entered into the "back" event of a Production Bar Chart.

```
event.source.parent.getComponent('Production Analysis Selector')
.prevDrillDown()
```

2.4.1.10 Production Analysis Selector

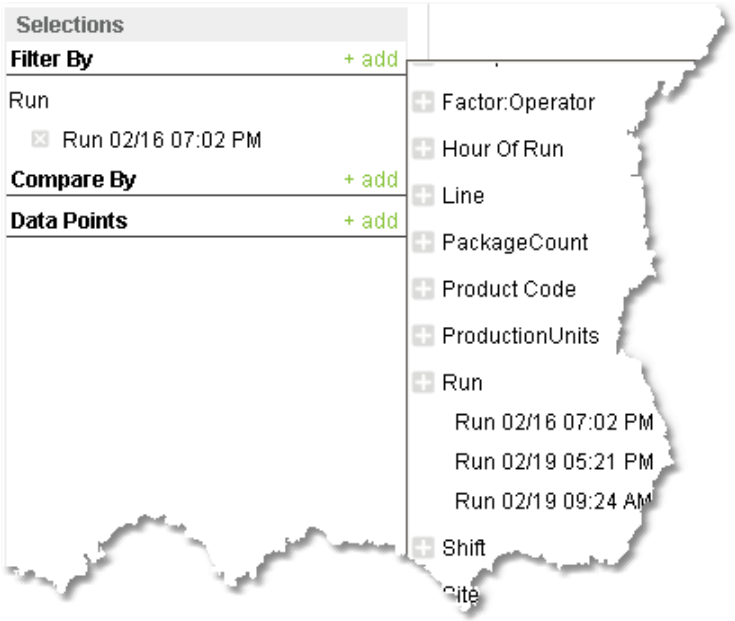
Description

A component that allows ad hoc selection of analysis data. As the user makes selections, this component will query the server for results. These results can be accessed through the Table Data, Chart Data and Line Chart Data properties to populate tables and charts.



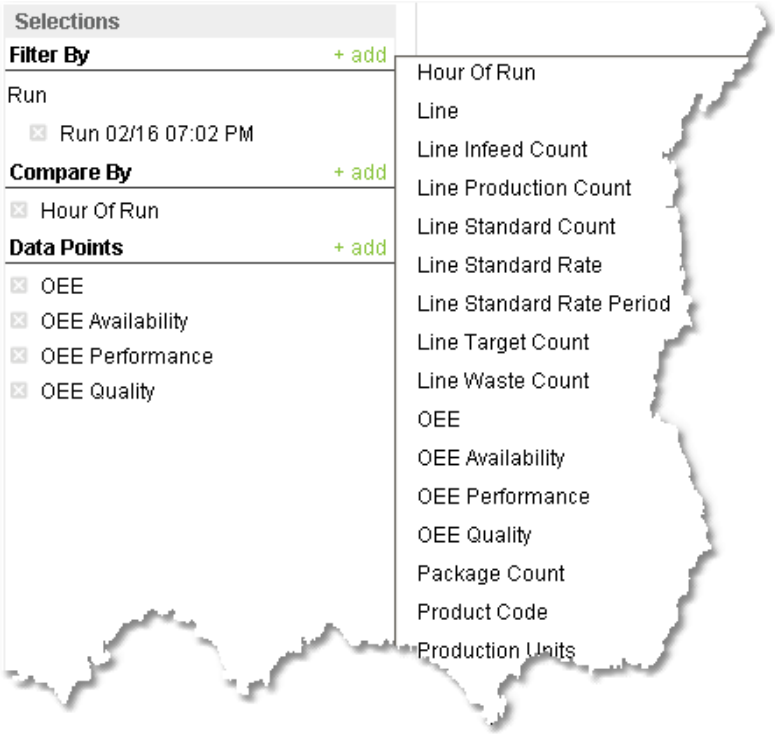
Production Analysis Selector

A filter can be added by selecting the + add link to the right of **Filter By**. A window panel will open and filter categories will be displayed. Click the ⊞ link by the filter category and specific filter items will be displayed. When selected they will be added to the filters as shown below. To minimize the number of filter options, only the options for the selected date range defined by the Start Date and End Date properties will be shown.



Filter By List

Compare By and Data Points work similarly to Filter By except there are no categories for these selections, just items.



Compare By and Data Points List

Selections can be removed by selecting the ☒ link to the left of the selection.

Properties

This component has standard Ignition properties with the addition of the following properties:

Table Data

This property holds data in a format that is optimized for binding to a table component.

```
Scripting name          tableData
Data Type               Dataset
```

Chart Data

This property holds data in a format that is optimized for binding to pie and bar chart components such as the Production Bar Chart and Production Pie Chart.

```
Scripting name          chartData
Data Type               Dataset
```

Line Chart Data

This property holds data in a format that is optimized for binding to a *line* chart component.

```
Scripting name          lineChartData
Data Type               Dataset
```

Drill Down Options

This property holds the drill down options appropriate for the current filter and compare by settings.

```
Scripting name          drillDownOptions
Data Type               Dataset
```

Previous Drill Down Enabled

This property indicates if there are entries in the drill down cache maintained by this component.

```
Scripting name          previousDrillDownEnabled
Data Type               Boolean
```

Provider

This property holds the current provider of analysis data. See Analysis Providers for available options.

```
Scripting name          provider
Data Type               String
```

Start Date

This property is the starting date for retrieving analysis data and determining available filter and compare by options.

```
Scripting name          startDate
Data Type               Date
```

End Date

This property is the ending date for retrieving analysis data and determining available filter and compare by options.

```
Scripting name          endDate
Data Type               Date
```

Filter Selection Summary

This property holds the current filter item selections to filter the analysis results by. If more than one item exists, they are separated by commas.

```
Scripting name          filterSummary
Data Type               String
```

Comparisons Selection Summary This property holds the current compare by item selections to group the analysis results by. If more than one item exists, they are separated by commas.

```
Scripting name          comparisonsSummary
Data Type               String
```

Data Points Selection Summary This property holds the currently selected data points to include in the results. If more than one item exists, they are separated by commas.

```
Scripting name          dataPointsSummary
Data Type               String
```

Data Format This property specifies the type of data to return from the server.

Options:

Table - Only data optimized for tables will be included in the results.
Chart - Only data optimized for charts will be included in the results.
Both - Table and chart data will be included in the results.

```
Scripting name          dataFormat
Data Type               AnalysisDataFormat
values                  Table
                        Chart
                        Both
```

Events

This component has standard Ignition events.

Methods

`drillDown(drillDownName, item)`

Sets all the analysis selections to new state dictated by the drill down definition.

parameters

drillDownName	A drill down definition name. This is typically supplied by the drill down event of one of the display components `Data Type` String
item	A drill down category. This is typically supplied by the drill down event of one of the display components `Data Type` Object

returns nothing

`prevDrillDown()`

parameters (none)
returns nothing

Example Code

This script would be entered into the "drillDown" event of a Production Bar Chart.

```
event.source.parent.getComponent('Production Analysis Selector')
.drillDown(event.getDrillDownName(), event.getCategory())
```

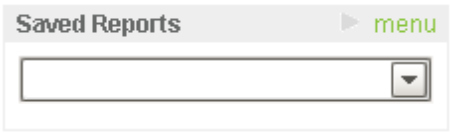
This script would be entered into the "back" event of a Production Bar Chart.

```
event.source.parent.getComponent('Production Analysis Selector')
.prevDrillDown()
```

2.4.1.11 Production Stored Analysis Selector

Description

A component that allows creating, recalling and saving analysis data selections in the Production Analysis Selector. This component will automatically use the available Production Analysis Selector in the container.

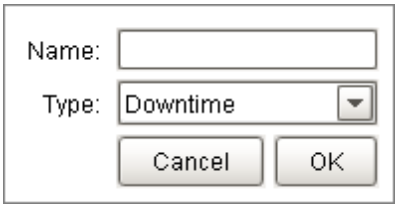


Stored Analysis Selector

By clicking on the

link, a menu with the option to create new, save, delete and rename analysis will popup.

To add a new stored analysis, click on **New** menu item, enter a name, select a type and click **OK**. This will create an empty analysis. Now the user can make filter, compare by and data point selections that will be saved and can easily be selected at a later time.



New Stored Analysis

To rename a new stored analysis, click on **Rename** menu item, enter a new name and click **OK**.

Rename Stored Analysis

To delete a stored analysis, click on **Delete** menu item, and select **Yes** to the confirmation message.

If changes to an analysis setting have been made and the user selects a different stored analysis, they will be prompted to save the changes. Alternatively, the changes can be saved by clicking on the **Save** menu item.

Properties
This component has standard Ignition properties.

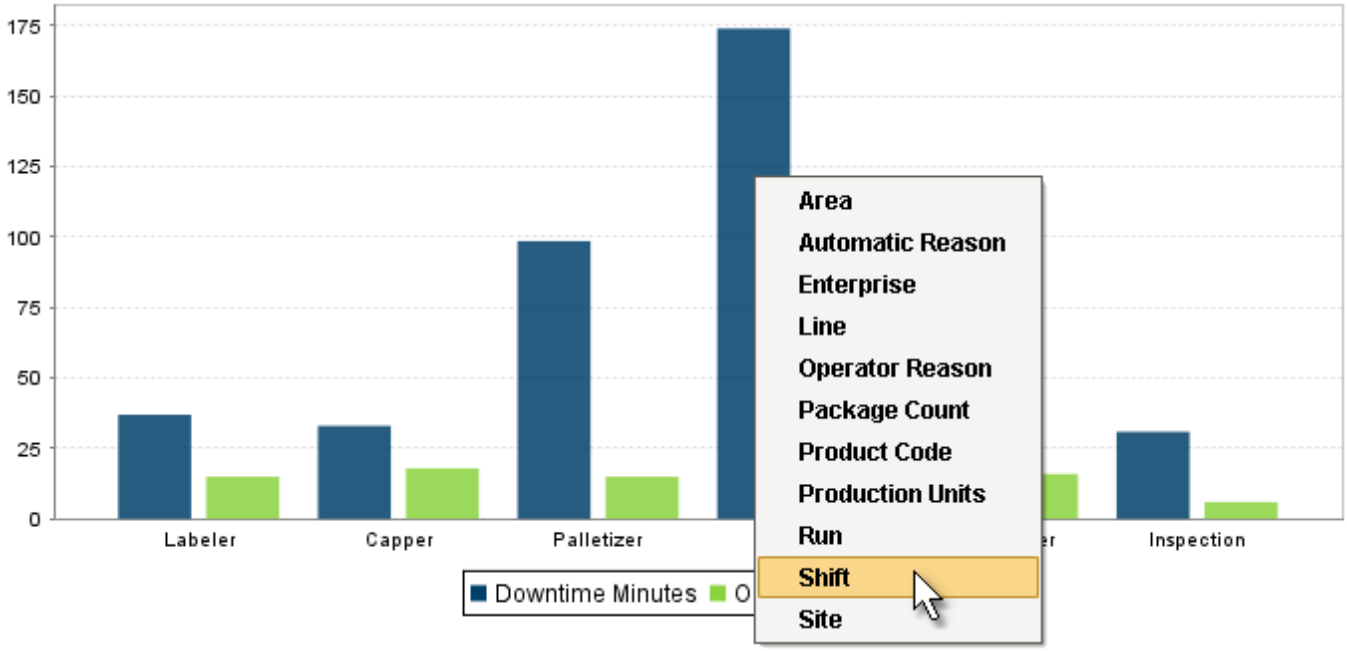
Events
This component has standard Ignition events.

Methods
(none)

2.4.1.12 Production Bar Chart

Description
A component that displays a pie chart with drill down capabilities. This extends from the Bar Chart Component that comes with Ignition.

When the user clicks on a bar of the bar chart, the drill down menu will appear. When an item in the drill down menu is clicked on, the drillDown event is fired. Script in the drillDown event is responsible for updating the Data property to change the results shown in the bar chart.The drill down menu information is set through the Drill Down Options property. The Drill Down Options can populated from the Analysis Controller, Analysis Selector, SQL Query, scripting or it can be manually defined in the designer.



Production Bar Chart

Properties

This component has the same properties as the Ignition Bar Chart Component with the addition of the following properties:

Drill Down Options

This is a Dataset that must have at least one column. The first column must be a data type of string. The values in the first column will be shown in the drill down options menu.
Typically, this property binds to the drill down options property in a Production Analysis Selector component.

```
Scripting name          drillDownOptions
Data Type               Dataset
```

Previous Drill Down Enabled

This controls the visibility of the "Back" drill down menu option. If it is set to true, "Back" will appear at the top of the drill down options.

```
Scripting name          previousDrillDownEnabled
Data Type               Boolean
```

Events

This component has the same events as the Ignition Pie Chart Component with the addition of the following events:

drillDown

Is fired when drill down menu item is selected. Excludes the "Back" menu item.

Event Properties

event.getDrillDownName() — Returns the text of selected drill down option menu item.
Data Type String

event.getCategory() — Returns the bar chart category that was clicked on to display the drill down menu. This is typically the first column of the Data property dataset.
Data Type Object

back

Event Properties — (none)

Methods

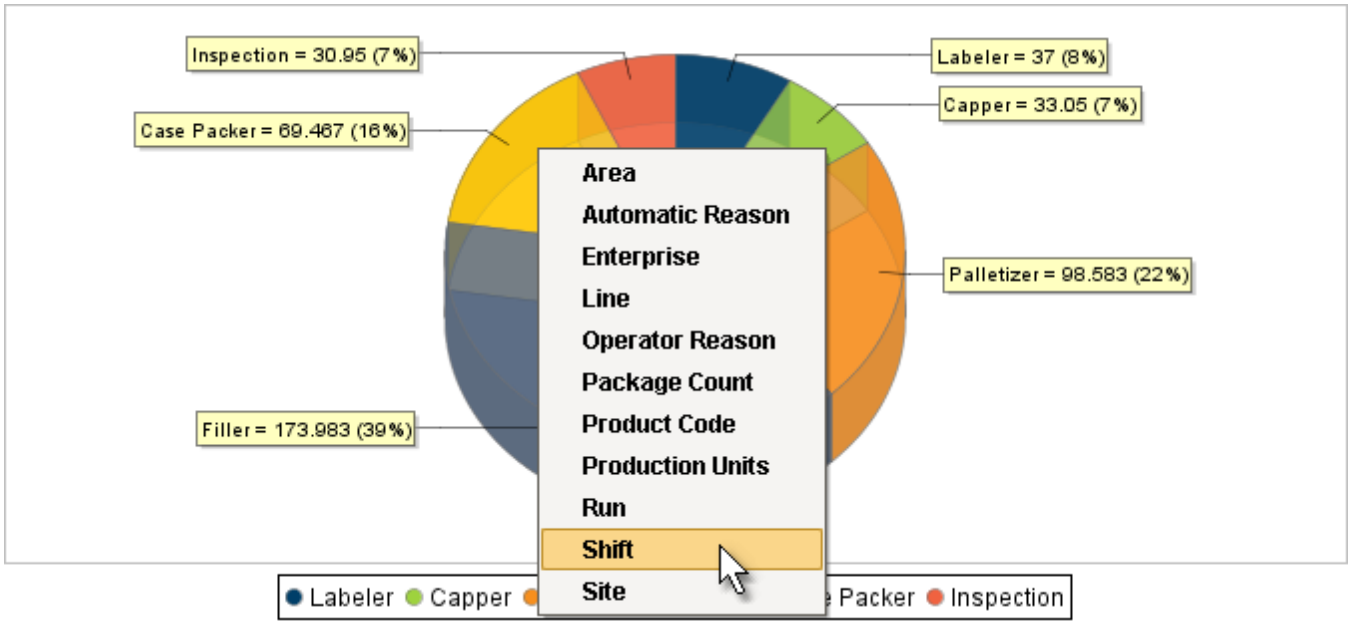
(none)

2.4.1.13 Production Pie Chart

Description

A component that displays a pie chart with drill down capabilities. This extends from the Pie Chart Component that comes with Ignition.

When the user clicks on a segment of the pie chart, the drill down menu will appear. When an item in the drill down menu is clicked on, the drillDown event is fired. Script in the drillDown event is responsible for updating the Data property to change the results shown in the pie chart. The drill down menu information is set through the Drill Down Options property. The Drill Down Options can populated from the Analysis Controller, Analysis Selector, SQL Query, scripting, or it can be manually defined in the designer.



Production Pie Chart

Properties

This component has the same properties as the Ignition Pie Chart Component with the addition of the following properties:

Drill Down Options

This is a Dataset that must have at least one column. The first column must be a data type of string. The values in the first column will be shown in the drill down options menu.
Typically, this property binds to the drill down options property in a Production Analysis Selector component.

```
Scripting name     drillDownOptions
Data Type          Dataset
```

Previous Drill Down Enabled

This controls the visibility of the "Back" drill down menu option. If it is set to true, "Back" will appear at the top of the drill down options.

```
Scripting name     previousDrillDownEnabled
Data Type          Boolean
```

© Inductive Automation

Events

This component has the same events as the Ignition Pie Chart Component with the addition of the following events:

drillDown — Is fired when drill down menu item is selected. Excludes the "Back" menu item.

Event Properties

event.getDrillDownName()	Returns the text of selected drill down option menu item. Data Type String
event.getCategory()	Returns the pie chart category that was clicked on to display the drill down menu. This is typically the first column of the Data property dataset. Data Type Object

back

Event Properties — (none)

Methods

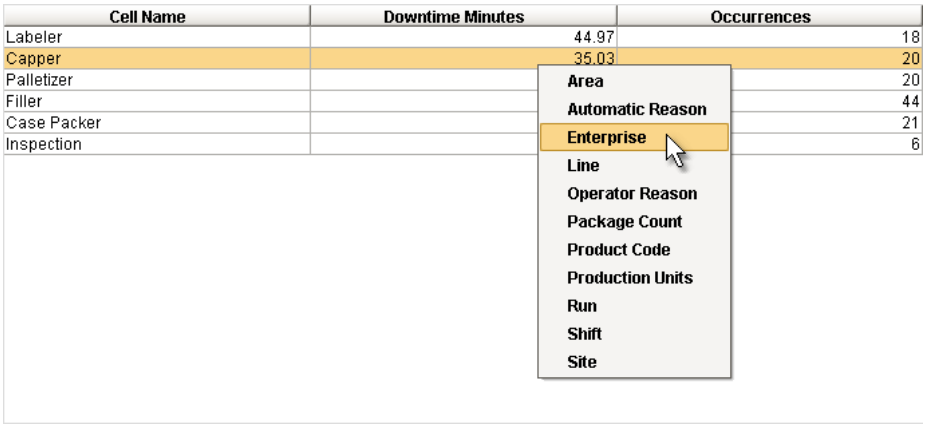
(none)

2.4.1.14 Analysis Table

Description

A component that displays tabular data with drill down capabilities. This extends from the Table Component that comes with Ignition.

When the user clicks on a row in the table, the drill down menu will appear. When an item in the drill down menu is clicked on, the drillDown event is fired. Script in the drillDown event is responsible for updating the Data property to change the results shown in the table.The drill down menu information is set through the Drill Down Options property. The Drill Down Options can populated from the Analysis Controller, Analysis Selector, SQL Query, scripting, or it can be manually defined in the designer.



Analysis Table

Properties

This component has the same properties as the Ignition Table Component with the addition of the following properties:

Allow Export
This controls the visibility of the "Export" menu option. If it is set to true, "Export" will appear at the top of the drill down options allowing the user to export the data appearing the table.
`Scripting name` `allowExport`
`Data Type` `Boolean`

Drill Down Options
This is a Dataset that must have at least one column. The first column must be a data type of string. The values in the first column will be shown in the drill down options menu.
Typically, this property binds to the drill down options property in a Production Analysis Selector component.
`Scripting name` `drillDownOptions`
`Data Type` `Dataset`

Previous Drill Down Enabled
This controls the visibility of the "Back" drill down menu option. If it is set to true, "Back" will appear at the top of the drill down options.
`Scripting name` `previousDrillDownEnabled`
`Data Type` `Boolean`

Events

This control has the same events as the Ignition Table Component with the addition of the following events:

drillDown
Is fired when drill down menu item is selected. Excludes the "Back" menu item.

Event Properties

event.getDrillDownName()	Returns the text of selected drill down option menu item. Data Type String
event.getCategory()	Returns the value of first column for the selected row. Data Type Object

back

Event Properties
(none)

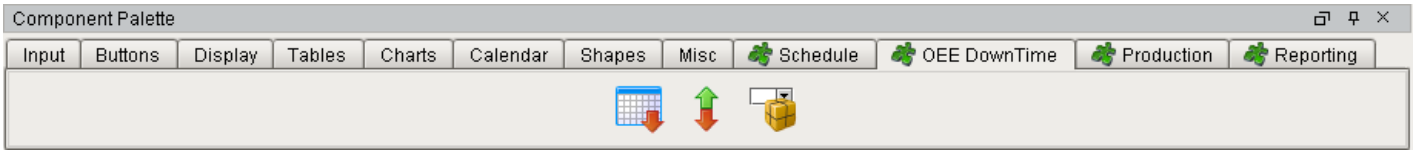
Methods

(none)

© Inductive Automation

2.4.2 Down Time Components

When the OEE Downtime Module, which is part of the OEE Downtime and Scheduling Module, is opened, a new component tab will appear. On it are components that provide functionality specific to the downtime and efficiency.

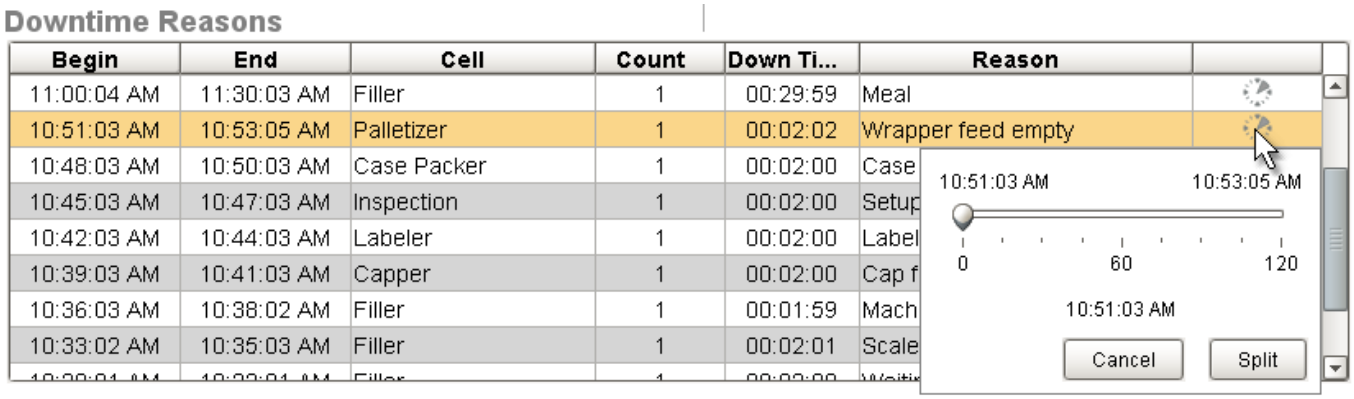


Down Time Components

2.4.2.1 Down Time Table

Description

A component that displays automatic downtime events for an active production run and allows the operator to select more specific downtime reasons for the event. It also allows the operator to split downtime events. This accommodates downtime events that have multiple reasons. For example, if a production *line* goes down because of a mechanical failure and when maintenance finishes the repair, it is time for break. The operator can split the downtime event into two events. One for mechanical failure and the other for break.

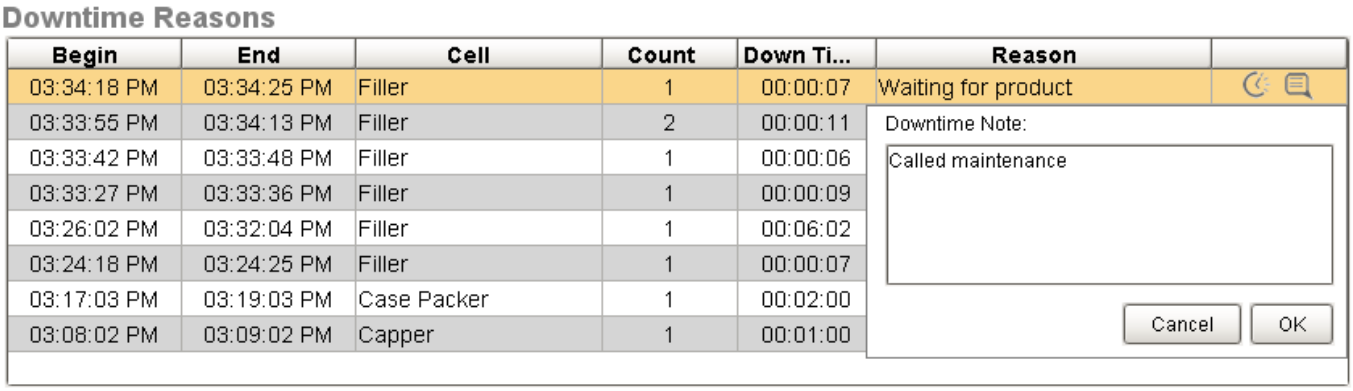


Splitting Down Time Reason

When the user clicks on the icon in the right-hand column, the downtime event split panel appears. The user can drag the time selector to the desired number of hours, minutes and seconds to split the event at. After the user clicks the **Split** button, two entries in the Down Time Table will appear with the exact same downtime reasons. The user can now select different downtime reasons for each entry.

When multiple downtime events occur for the same automatically detected downtime reason, they will be combined into a single entry. The **Count** column will indicate the number of events and the **Downtime** column will reflect the total downtime of the combined events. The **Begin** column will be the start of first occurrence and the **End** column will be the end of the last occurrence. The user can click on the icon to separate the combined downtime events. This allows selecting different downtime reasons for each of the downtime events.

© Inductive Automation



Commenting on Down Time Reason

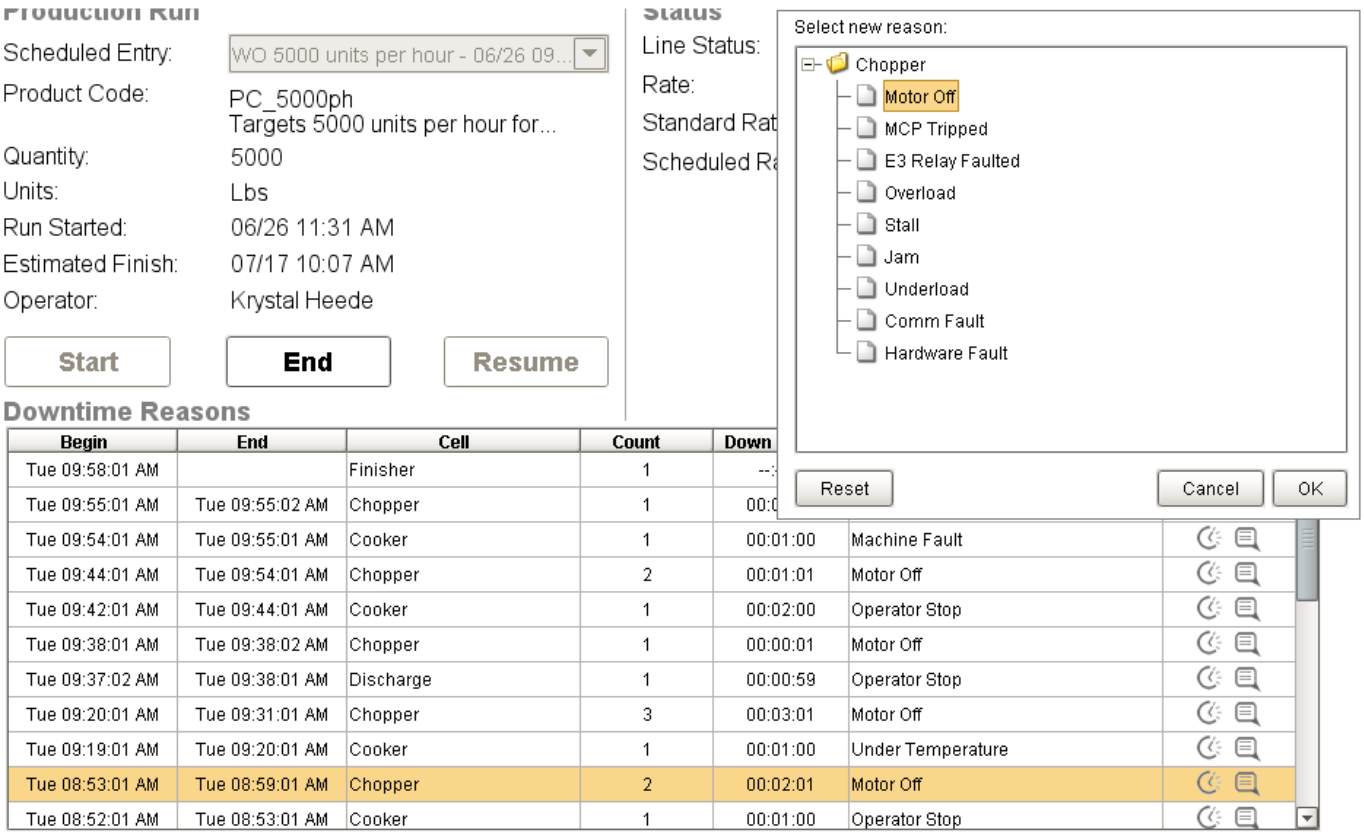
When the use clicks on the icon in the right-hand column, the downtime note panel appears. The user can enter a note that will be associated with the downtime reason entry.

Editing Reasons

If the **Editable** property is set to true then the user may change the original reason generated to a more appropriate reason. This is useful if, for instance, the machine can only generate a general fault code and the user needs to enter in a specific fault after determining the cause.
The reasons that are available for selection are controlled by the configuration of the Downtime Reasons and the **Reason Selection Method** property of the table.
The selectable reasons will be displayed in a pop-up panel tree view type control when the reason column is clicked on the desired row. The reason tree view layout can be controlled by several properties (see property description below) such as height, width, font and icons for selectable and non-selectable reasons.

When editing a reason with the Reason Selection Method set to "Original Cell" the user will presented with a tree view of the appropriate reasons for the original cell.

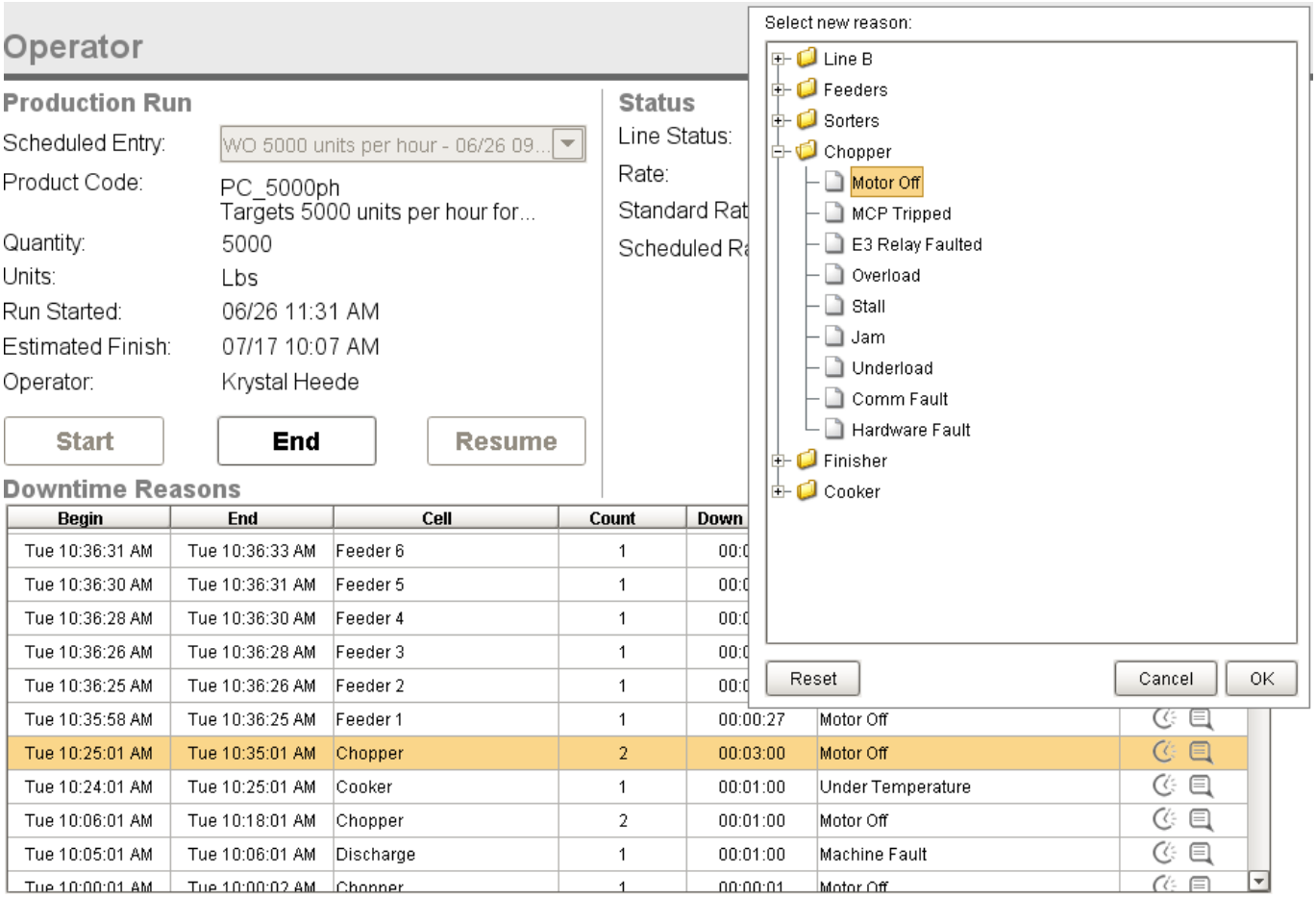
© Inductive Automation



Editing a reason for Original Cell

When editing a reason with the Reason Selection Method set to "Any Cell" the user will presented with a tree view of all cells and reasons. This allows the user to select a more appropriate cell and reason than what was originally detected by the system.

© Inductive Automation



Editing a reason for Any Cell

Original Reason and Cell retention

The original Reason, Cell and Cell Group (if applicable) are never lost when a user edits a reason. The reason selector pop-up panel allows the user to reset to the original reason by selecting the "Reset" button on the left of the panel. Also note that the analysis components have datapoints defined for accessing the original reason.

Using the Table Customizer

A table customizer is available by right clicking the down time table in the designer and selecting "Customizers" -> "Table Customizer". It is similar to the table customizer in a standard Ignition table but the fields are pre-defined for the downtime table component.

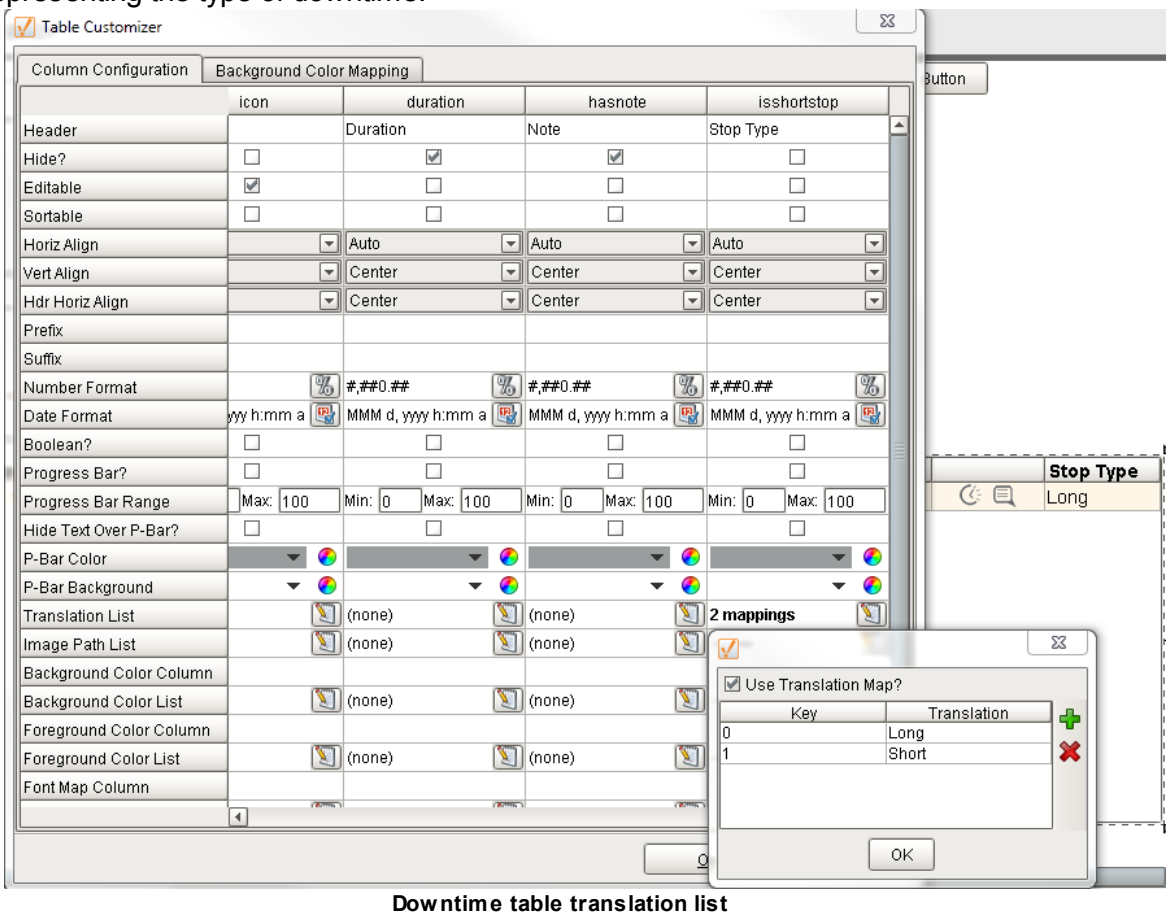
Field Name	Description	Comments
begintime	When the event started	
endtime	When the event ended	
cellid	Original Cell ID	
cellname	Original Cell Name	
operatorcellid	Operator selected Cell ID	
operatorcellname	Operator Selected Cell Name	
count	Event Count	Indicates the number of times this event has occured consecutively

© Inductive Automation

Field Name	Description	Comments
totalseconds	Down Time	event duration formatted as a string in hours, minutes and seconds.
reasoncode	Operator Selected Reason code	
description	Reason Description	The selected reason code description.
ids	Internal ID's	Used by the system.
linestate	Original Reason Code	
icon	Split and note icons	Used by the system.
duration	Duration	This is the elapsed seconds of this event. Since it is an integer it can be used for background mapping, etc.
hasnote	Indicates this row has a note associated with it.	
isshortstop	Indicates if this reason is short or long.	Will indicate 0 for long stops and 1 for short.

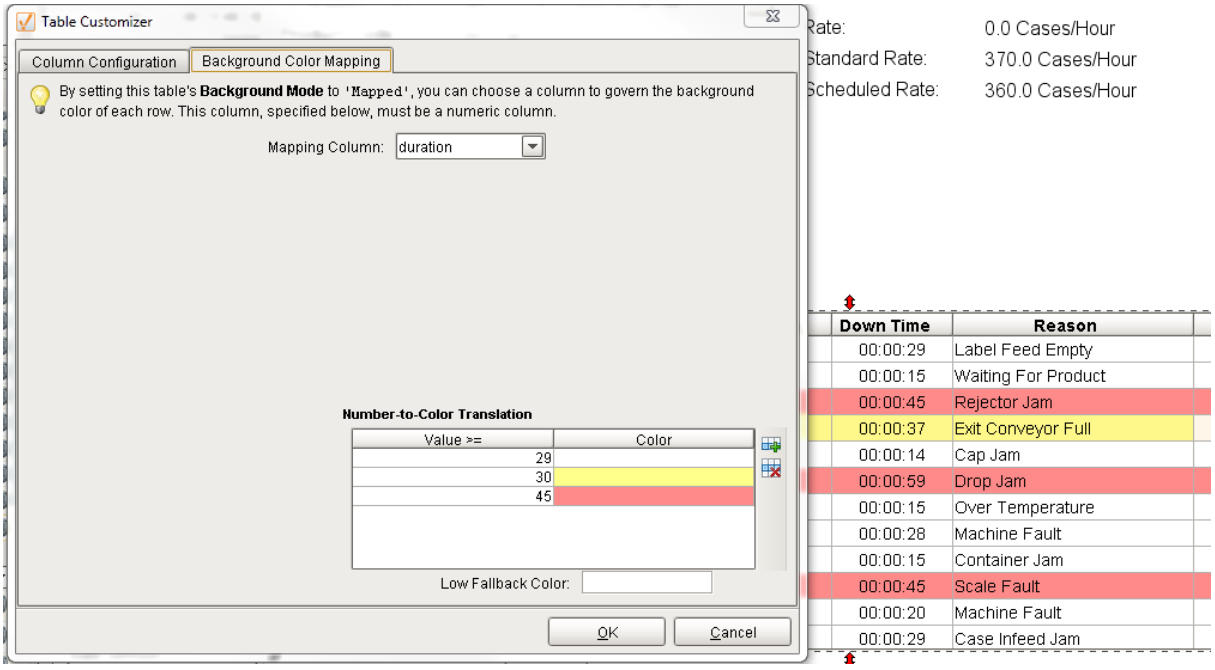
Downtime table customizer fields

Example using Translation List: In the "isshortstop" column configuration if you create a translation list and set 0=Long and 1=Short and set the field to not be hidden you will see a column with either Long or Short representing the type of downtime.



Downtime table translation list

Example using Background Mapping: Select the duration column as the mapping column. Set the translation for the seconds duration. Any value up to 29 seconds will have a white background, any value from 30 to 44 seconds will have a yellow background and any value greater than or equal to 45 will have a red background. Make sure to set the "Background Mode" property of the downtime table to "Mapped".



Properties

This component has standard Ignition events with the addition of the following properties:

Line Path

The *line* path of the production *line* that this component is associated with. This is the full path name of the *line* starting with the project name.

For example: "OEEDemo\Your Enterprise\Your Site\Your Area\Line 1"

```
Scripting name          linePath
Data Type               String
```

Editable

This controls if users can change reason codes and split downtime events.

```
Scripting name          editable
Data Type               Boolean
```

Enable Notes

If true users can enter notes for each downtime entry in the table.

```
Scripting name          enableNotes
Data Type               Boolean
```

Activity Timeout

Indicates the number of seconds the table will "freeze" after any user activity, such as scrolling, is performed on the table. Keeps the table from immediately jumping to a new event until the timeout is reached.

```
Scripting name          activityTimeout
Data Type               Integer
```

© Inductive Automation

Reason Selection Method

Controls the ability to select reasons from other than the originating cell.

```
Scripting name          cellSelectionType
Data Type               Integer
Values                  Original Cell = 0
                        Any Cell = 1
```

Run ID

Enter the run id when the downtime table is used to view previous runs. -1 indicates the current run. Normally is linked to the "Line Run Selector" component to get a previous run id.

Note: When set to a value other than -1 the table will not be notified of new events even if the runid is set to the current run id.

```
Scripting name          runid
Data Type               Integer
```

Reason Tree Expand sub reasons

When true sub reasons will be automatically expanded under the current reason when displaying the selection panel.

```
Scripting name          autoExpandSubReasons
Data Type               Boolean
```

Reason Tree Width

The width of the pop-up reason tree panel.

```
Scripting name          reasonTreeWidth
Data Type               Integer
```

Reason Tree Height

The height of the pop-up reason tree panel.

```
Scripting name          reasonTreeHeight
Data Type               Integer
```

Reason Tree Row Height

Controls the height of each row in the reason tree view. The increased space is useful when touchscreen mode is being used.

```
Scripting name          reasonTreeRowHeight
Data Type               Integer
```

Reason Tree Font

The font of the pop-up reason tree panel.

```
Scripting name          reasonTreeFont
Data Type               Integer
```

Reason Tree Scroll Bar Width

The scroll bar width of the tree view. Useful when touchscreen mode is being used.

```
Scripting name          reasonTreeScrollBarWidth
Data Type               Integer
```

Reason Tree Button Size %

The size of the buttons for the selection in the reason pop-up panel. Useful when touchscreen mode is being used.

```
Scripting name          reasonTreeButtonSizePct
Data Type               Integer
```

Show the Reason Reset Button

When set to false the reset button will not be available in the reason pop-up panel.

```
Scripting name          showResetReasonButton
Data Type               Boolean
```

Selectable Reason Icon Path

The icon to use for reasons that are selectable.

```
Scripting name          selectableReasoniconpath
Data Type               String
```

Folder Icon Path

The icon to use for items in the reason pop-up panel that are not selectable, such as a cell.

```
Scripting name          foldericonpath
Data Type               String
```

Display Filter Type

For long runs that span shifts and/or days controls how many shifts will be shown. If there are numerous downtime events settings other than "Current Shift" can cause delay in the downtime table.

```
Scripting name          shiftDisplayType
Data Type               Integer
Values                  Current Shift = 0
                        Previous and Current Shift = 1
                        Selected Shift Sequence = 2
                        Display Cutoff Duration (Minutes) = 3
                        All = 4
```

Selected Shift Sequence

This is the shift sequence to start displaying downtime events if the "Number of shifts to display" property is set to "Selected Shift Sequence". Shift sequence numbers are the consecutive numbers of shifts of a run.

```
Scripting name          selectedShiftSequence
Data Type               Integer
```

Display Cutoff Duration (Minutes)

Displays downtime events where the end time is within this number of minutes from the current time. Applicable when Display Filter Type is set to Display Cutoff Duration (Minutes).

```
Scripting name          dutoffMinutes
Data Type               Integer
```

Stop FilterType

Determines if the table will display long, short or both types of downtime events.

```
Scripting name          selectedStopType
Data Type               Integer
Values                  Both = 0
                        Short Stops = 1
                        Long Stops = 2
```

Rollup Interval (Seconds)

Rolls up events that are consecutive, identical and occured within this many seconds of each other. If set to less than 1 then no rollup will occur.

```
Scripting name          rollupInterval
Data Type               Integer
```

© Inductive Automation

Events

This component has standard Ignition Table events and the following custom events:

reasonUpdated Is fired after a user has updated an existing reason.

Methods

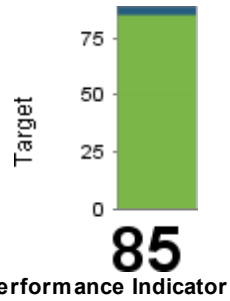
(none)

© Inductive Automation

2.4.2.2 Performance Indicator

Description

A component that displays an indication of actual versus target values. It provides a visual indication to users that is easy to comprehend with a quick glance. These values can be unit count, OEE or any values residing in SQLTags.



This is similar to a bar chart except that it only has 2 series or bars. Also, the values reside in SQLTags instead of having to setup values in an Ignition Dataset.

Properties

This component has the same properties as the Ignition Bar Chart Component with the addition of the following properties:

Actual Value The value that is represented by the actual indication bar.

```
Scripting name     actualValue
Data Type          Double
```

Actual Label The text displayed to describe the actual value.

```
Scripting name     actualLabel
Data Type          String
```

Actual Series Color The color to use for the actual indication bar.

```
Scripting name     actualSeriesColor
Data Type          Color
```

Chart Type The type of chart to show.

```
Scripting name     chartType
Data Type          CategoryItemRenderer
Options:           3D Bars
                   3D Stacked Bars
                   Area
                   Bars
                   Layered
                   Stacked Bars
                   Indicator
```

© Inductive Automation

Target Value The value represented by the target indication bar.

```
Scripting name     targetValue
Data Type          Double
```

Target Label The text displayed to describe the target value.

```
Scripting name     targetLabel
Data Type          String
```

Target Series Color The color to use for the target indication bar.

```
Scripting name     targetSeriesColor
Data Type          Color
```

Editable This controls if users can change reason codes and split downtime events.

```
Scripting name     editable
Data Type          Boolean
```

Events

This component has standard Ignition events

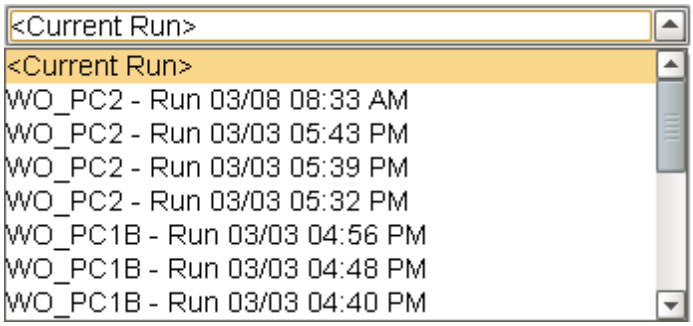
Methods

(none)

2.4.2.3 Line Run Selector

Description

A component that provides users to select a production run from a drop-down list of available runs on a production *line*. The user can also select the current run by selecting <Current Run>.



Line Run Selector

Properties

This component has standard Ignition properties with the addition of the following properties:

Line Path
The *line* path of the production *line* that this component is associated with. This is the full path name of the *line* starting with the project name.

For example: "OEEDemo\Your Enterprise\Your Site\Your Area\Line 1"

```
Scripting name     linePath
Data Type          String
```

Include Work Order Description
When true displays the work order description in the drop down list (if available).

```
Scripting name     includeDescription
Data Type          Boolean
```

Run ID
The currently selected production run ID. This is the ID for the "Run" database table.

```
Scripting name     runID
Data Type          Integer
```

From Date
The beginning date to select runs from.

```
Scripting name     fromDate
Data Type          Date
```

To Date
The ending date to select runs from.

```
Scripting name     toDate
Data Type          Date
```

Events

This component has standard Ignition events.

Methods

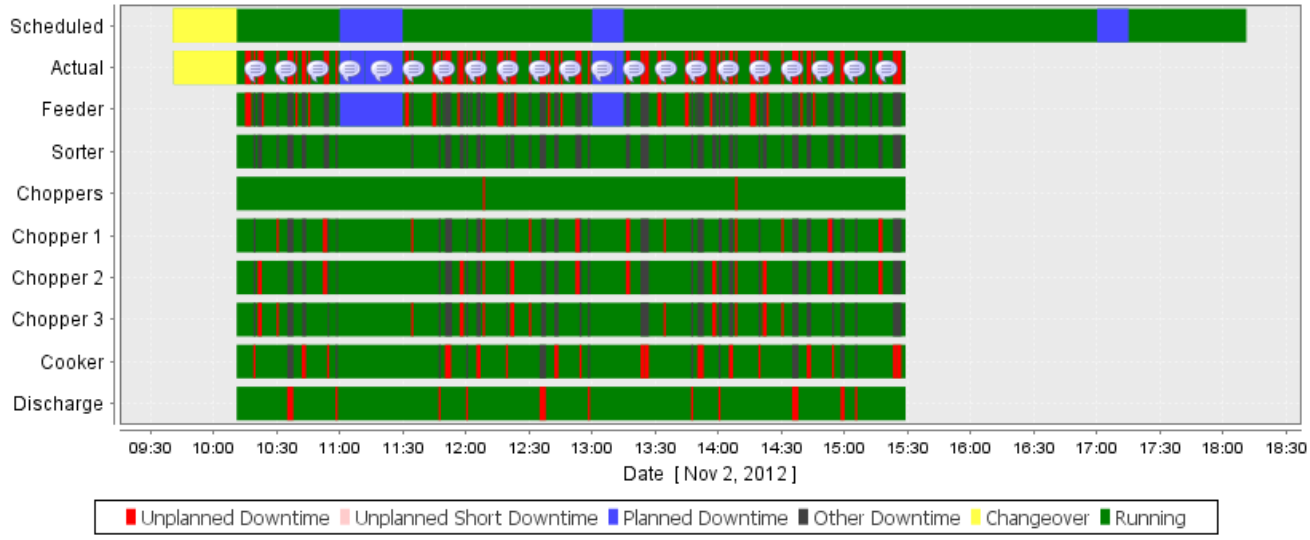
(none)

© Inductive Automation

2.4.2.4 Analysis Time Chart

Description

A component that displays the line and cell downtime events of a run in a visual time chart.

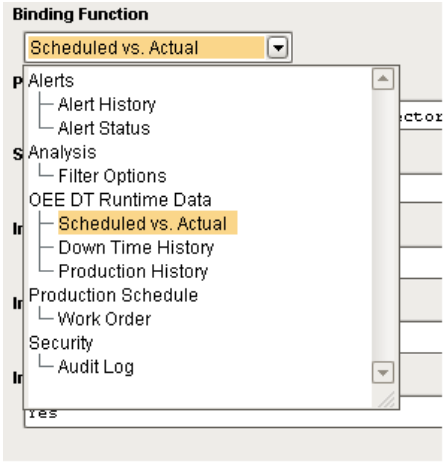


Analysis Time Chart

Properties

This component has standard Ignition properties with the addition of the following properties:

Series Data Controls the data that is displayed in the analysis time chart and needs to be set to the Schedule vs. Actual binding function.



See Scheduled vs. Actual in the Binding Function Reference section.

```
Scripting name     data
Data Type          Dataset
```

Duration Text

The text to show for the duration values in the info popup panel.

```
Scripting name     durationText
Data Type          String
```

Enable Info Popup

Show an info popup when you right click on the chart.

```
Scripting name     enableInfoPopup
Data Type          Boolean
```

Run Bar Color

Selects the color of the running section of the chart.

```
Scripting name     runColor
Data Type          Color
```

Run Legend Text

Set the text that will appear in the legend that represents the running section of the chart.

```
Scripting name     runLegend
Data Type          String
```

Changeover Bar Color

Selects the color of the changeover section of the chart.

```
Scripting name     changeoverColor
Data Type          Color
```

Changeover Legend Text

Set the text that will appear in the legend that represents the changeover section of the chart.

```
Scripting name     changeoverLegend
Data Type          String
```

Planned Downtime Bar Color

Selects the color of the planned downtime section of the chart.

```
Scripting name     plannedDowntimeColor
Data Type          Color
```

Planned Downtime Legend Text

Set the text that will appear in the legend that represents the planned downtime section of the chart.

```
Scripting name     plannedLegend
Data Type          String
```

Unplanned Downtime Bar Color

Selects the color of the unplanned downtime section of the chart.

```
Scripting name     unplannedDowntimeColor
Data Type          Color
```

Unplanned Downtime Legend Text

Set the text that will appear in the legend that represents the unplanned downtime section of the chart.

```
Scripting name     unplannedLegend
Data Type          String
```

© Inductive Automation

Unplanned Short Downtime Bar Color Selects the color of the unplanned short downtime section of the chart.

```
Scripting name     unplannedShortDowntimeColor
Data Type          Color
```

Unplanned Short Downtime Legend Text Set the text that will appear in the legend that represents the unplanned short downtime section of the chart.

```
Scripting name     unplannedShortLegend
Data Type          String
```

Events

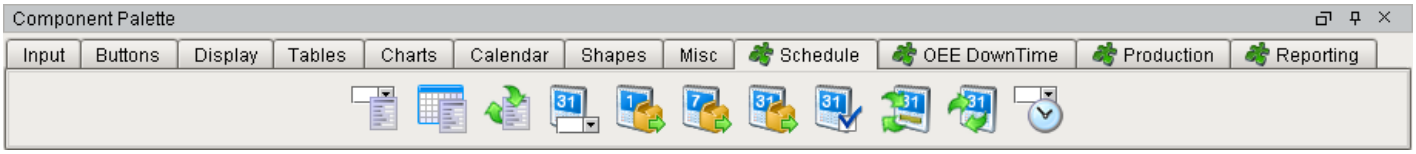
This component has standard Ignition events.

Methods

(none)

2.4.3 Schedule Components

When the Schedule Module, which is part of the OEE Downtime and Scheduling Module, a new component tab will appear. On it are components that provide functionality specific to the work orders, product codes and scheduling.

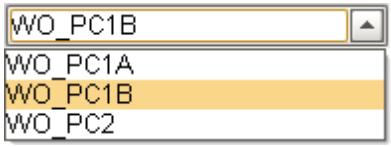


Schedule Components

2.4.3.1 Work Order Selector

Description

A component that provides users to select a work order from a drop-down list of available work orders for a production *line*. The available options include only work orders for product codes that are enabled to run on the specified production *line*. All work orders are automatically displayed from the "WorkOrder" database table without the need for custom SQL statements or script.



Work Order Selector

Properties

This component has standard Ignition properties with the addition of the following properties:

Line Path

The *line* path of the production *line* that this component is associated with. This is the full path name of the *line* starting with the project name.

For example: "OEEDemo\Your Enterprise\Your Site\Your Area\Line 1"

```
Scripting name     linePath
Data Type          String
```

Selected Work Order ID

The currently selected work order ID. This is the ID for the "WorkOrder" database table.

```
Scripting name     selectedWorkOrderID
Data Type          Integer
```

Events

This component has standard Ignition events.

Methods

(none)

2.4.3.2 Work Order Table

Description

A component that displays all the available work orders in a table and calculates the units produced, scheduled and remaining for each work order. All work orders are automatically displayed from the "WorkOrder" database table within the date range of **From Date** and **To Date** properties without the need for custom SQL statements or script.

Closed	Hide	Work Order	Product Code	Description	Quantity	Produced	Scheduled	Remaining
☐	☐	WO_PC1B	PC_001	Product Code 1	42,000	43,889	480	-2,369
☐	☐	WO_PC2	PC_002	Product Code 2	67,000	30,747	1,540	34,713
☐	☐	WO_PC3	PC_003	Product Code 3	623,852	0	0	623,852
☐	☐	WO_PC1A	PC_001	Product Code 1	10,000	32,488	0	-22,488

Work Order Table

The users can click on a checkbox in the **Closed** column to close out a work order. After it is closed out, it will no longer show in the Work Order Table component and it will not be available in any other work order selector components. This feature is provided because some production runs may finish before the target number of units are produced due to lack of raw materials, change in production priorities, etc.

The user can also click on a checkbox in the **Hide** column to hide the work order from being shown in the Work Order Component. Implementations that integrate with other software systems, such as an ERP system, may show work orders that are not relevant to this system. By hiding them, this list can be kept clean of unrelated work orders.

Properties

This component has standard Ignition properties with the addition of the following properties:

To Date This property is the starting date of when work orders were created.

```
Scripting name        startDate
Data Type             Date
```

From Date This property is the ending date of when work orders were created.

```
Scripting name        endDate
Data Type             Date
```

Show Closed If set to true, will show the closed work orders.

```
Scripting name        showClosed
Data Type             Boolean
```

Show Hidden If set to true, will show the hidden work orders.

```
Scripting name        showHidden
Data Type             Boolean
```

Events

This component has standard Ignition events.

Methods

(none)

2.4.3.3 Work Order Controller

Description

An invisible component that provides adding, editing and deleting work orders. The term *invisible component* means that the control appears during design time, but is not visible during runtime. Work orders are stored in the "WorkOrder" database table and this control handles all SQL statements, duplicate checking, etc.

Alternatively, work orders can added directly into the "WorkOrder" database table directly, bypassing the OEE Downtime and Scheduling Module. This method supports integration to ERP or other software systems.

Properties

This component has standard Ignition properties.

© Inductive Automation

Events

This component has standard Ignition events.

Methods

`addWorkOrderEntry(workOrder, productCode, quantity)`

Add new work order.

parameters

workOrder	The work order number to add to the database. Data Type	String
productCode	The product code to produce for work order being added. Data Type	String
quantity	The quantity of units to produce for work order being added. Data Type	Integer

returns

message	Contains a description of any error encountered, otherwise it will be empty Data Type	String

---

`editWorkOrderEntry(workOrder, productCode, quantity, workOrderID)`

Edit an existing work order.

parameters

workOrder	The work order number to add to the database. Data Type	String
productCode	The product code to produce for work order being added. Data Type	String
quantity	The quantity of units to produce for work order being added. Data Type	Integer
workOrderID	The ID of the work order to modify. This is the ID for the "WorkOrder" database table. Data Type	Integer

returns

message	Contains a description of any error encountered, otherwise it will be empty. Data Type	String

`deleteWorkOrderEntry(workOrderID)`

Delete an existing work order.

parameters

workOrderID	The ID of the work order to modify. This is the ID for the "WorkOrder" database table. Data Type Integer

returns

message	Contains a description of any error encountered, otherwise it will be empty. Data Type String

Example Code

The following script can be entered in a button's actionPerformed event. It will add the work order to the database. The return message will indicate if the there are any issues adding the product code, such as if the work order already exists.

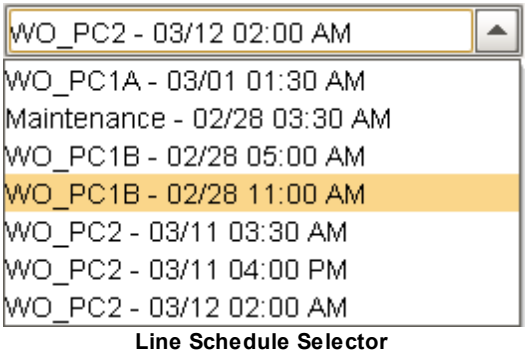
```
esp = event.source.parent # shorthand
workOrder = esp.getComponent('WorkOrderField').text
prodCode = esp.getComponent('ProductCodeDropdown').selectedStringValue
quantity = esp.getComponent('QuantityField').intValue

ctrl = esp.getComponent('Work Order Controller')
result = ctrl.addWorkOrderEntry(workOrder, prodCode, quantity)
if len(result) == 0:
    system.nav.closeParentWindow(event)
```

2.4.3.4 Line Schedule Selector

Description

A component that provides users to select a scheduled entry from a drop-down list of available schedule entries for a production *line*. The available options include only schedule entries that were scheduled for the production *line* and have not already been selected. All schedule entries are automatically displayed from the "Schedule" database table without the need for custom SQL statements or script.



Line Schedule Selector

Properties

This component has standard Ignition properties with the addition of the following properties:

Line Path The *line* path of the production *line* that this component is associated with. This is the full path name of the *line* starting with the project name.
For example: "OEEDemo\Your Enterprise\Your Site\Your Area\Line 1"
Scripting name linePath
Data Type String

Schedule ID The currently selected ID of the schedule entry. This is the ID for the "Schedule" database table.
Scripting name scheduleID
Data Type Integer

Include Closed Work Orders If true will allow selection of schedules that have not been run and are scheduled against a closed work order. Default is true.
Scripting name includeClosedWorkOrders
Data Type Boolean

Include Disabled Product Codes If true will allow selection of schedules that have not been run and are scheduled against a disabled product code. Default is true.
Scripting name includeDisabledProductCodes
Data Type Boolean

Events

This component has standard Ignition events.

Methods

`selectNextRun()`

Select next available schedule run.
parameters (none)
returns nothing

`selectRun(newScheduleID)`

Set new schedule ID.
parameters

newScheduleID	The ID of the schedule item to modify. This is the ID for the "Schedule" database table. Data Type Integer

returns

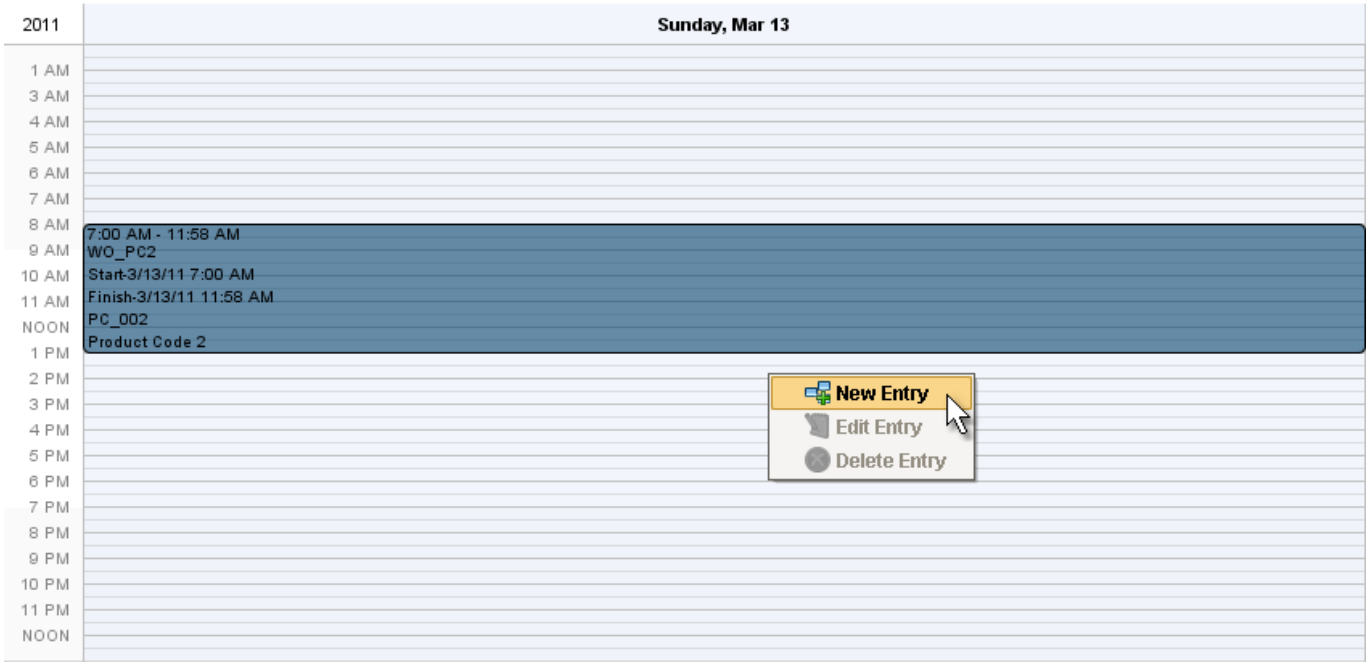
message	Contains a description of any error encountered, otherwise it will be empty Data Type String

2.4.3.5 Schedule Day View

Description

A component that displays scheduled entries for a selected day. This extends from the Day View Component that comes with Ignition to support adding, editing and deleting schedule entries. All schedule entries are automatically displayed from the "Schedule" and other database tables without the need for custom SQL statements or script.

When the user right clicks on a time, a popup menu will appear with options to add, edit or delete a schedule entry.



Schedule Day View

Properties

This component has the same properties as the Ignition Day View Component with the addition of the following properties:

Line Path The *line* path of the production *line* that this component is associated with. This is the full path name of the *line* starting with the project name.
For example: "OEEDemo\Your Enterprise\Your Site\Your Area\Line 1"

```
Scripting name     linePath
Data Type          String
```

Current Date The date to show schedule entries for.

```
Scripting name     currentDate
Data Type          Date
```

© Inductive Automation

Event Display Format Allows a custom string with replaceable parameters to be used. Leave blank to get the default event display string.

```
Scripting name     eventDisplayFormat
Data Type          String
parameters         {wo}  = Work Order
                   {co}  = Scheduled Change Over Start Date
                   {st}  = Scheduled Production Start Date
                   {fn}  = Scheduled Finish Time Date
                   {pc}  = Product Code
                   {pcd} = Product Code description
                   {qty} = Scheduled Quantity
                   {nt}  = Note
                   {eb}  = Entered By
                   |     = line feed (the pipe character)

Example:
If you set Event Display Format property to:
       Product Code = {pc} - {pcd}|Note: {nt}
The display string would look like:
       Product Code = PC243 - 24oz, 3 pack
       Note: Short run requested by customer
```

Events

This control has the same events as the Ignition Table Component with the addition of the following events:

newEvent Is fired when the **New Entry** menu item is selected.

editEvent Is fired when the **Edit Entry** menu item is selected.

deleteEvent Is fired when the **Delete Entry** menu item is selected.

Methods

(none)

Example Code

The following script can be entered into the newEvent event of this component. It collects the selected time when the right-click occurred and opens a new window with the collected values as parameters.

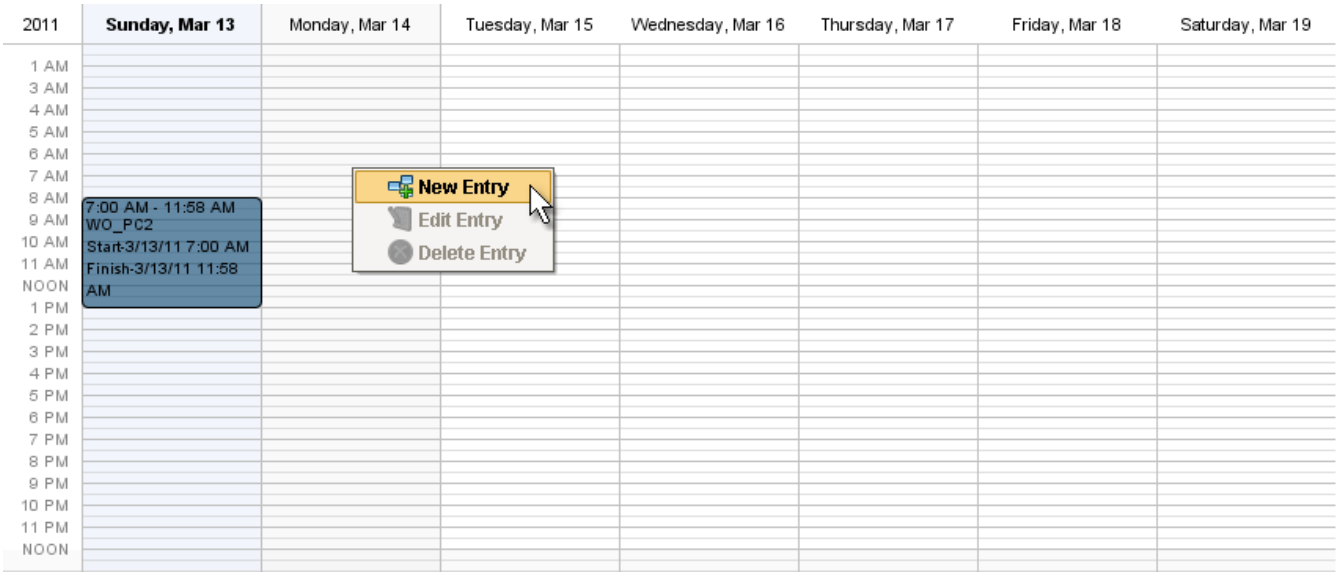
```
param1 = event.source.parent.getComponent('Production Line Selector').selectedLinePath
param2 = event.source.parent.getComponent('Schedule Day View').hoveredTimeLatched
system.nav.openWindow('ScheduleNew', {'LinePath' : param1, 'CurrentDate' : param2})
system.nav.centerWindow('ScheduleNew')
```

2.4.3.6 Schedule Week View

Description

A component that displays scheduled entries for a selected week. This extends from the Week View Component that comes with Ignition to support adding, editing and deleting schedule entries. All schedule entries are automatically displayed from the "Schedule" and other database tables without the need for custom SQL statements or script.

When the user right clicks on a time, a popup menu will appear with options to add, edit or delete a schedule entry.



Schedule Week View

Properties

This component has the same properties as the Ignition Week Component with the addition of the following properties:

Line Path — The *line* path of the production *line* that this component is associated with. This is the full path name of the *line* starting with the project name.
For example: "OEEDemo\Your Enterprise\Your Site\Your Area\Line 1"

```
Scripting name     linePath
Data Type          String
```

Current Date The date to show schedule entries for. The date can be any day from Sunday to Saturday.

```
Scripting name     currentDate
Data Type          Date
```

Event Display Format Allows a custom string with replaceable parameters to be used. Leave blank to get the default event display string.

```
Scripting name     eventDisplayFormat
Data Type          String
parameters         {wo}  = Work Order
                   {co}  = Scheduled Change Over Start Date
                   {st}  = Scheduled Production Start Date
                   {fn}  = Scheduled Finish Time Date
                   {pc}  = Product Code
                   {pcd} = Product Code description
                   {qty} = Scheduled Quantity
                   {nt}  = Note
                   {eb}  = Entered By
                   |     = line feed (the pipe character)

Example:
If you set Event Display Format property to:
       Product Code = {pc} - {pcd}|Note: {nt}
The display string would look like:
       Product Code = PC243 - 24oz, 3 pack
       Note: Short run requested by customer
```

Events

This control has the same events as the Ignition Table Component with the addition of the following events:

newEvent Is fired when the **New Entry** menu item is selected.

editEvent Is fired when the **Edit Entry** menu item is selected.

deleteEvent Is fired when the **Delete Entry** menu item is selected.

Methods

(none)

Example Code

The following script can be entered into the newEvent event of this component. It collects the selected time when the right-click occurred and opens a new window with the collected values as parameters.

```
param1 = event.source.parent.getComponent('Production Line Selector').selectedLinePath
param2 = event.source.parent.getComponent('Schedule Week View').hoveredTimeLatched
system.nav.openWindow('ScheduleNew', {'LinePath' : param1, 'CurrentDate' : param2})
system.nav.centerWindow('ScheduleNew')
```

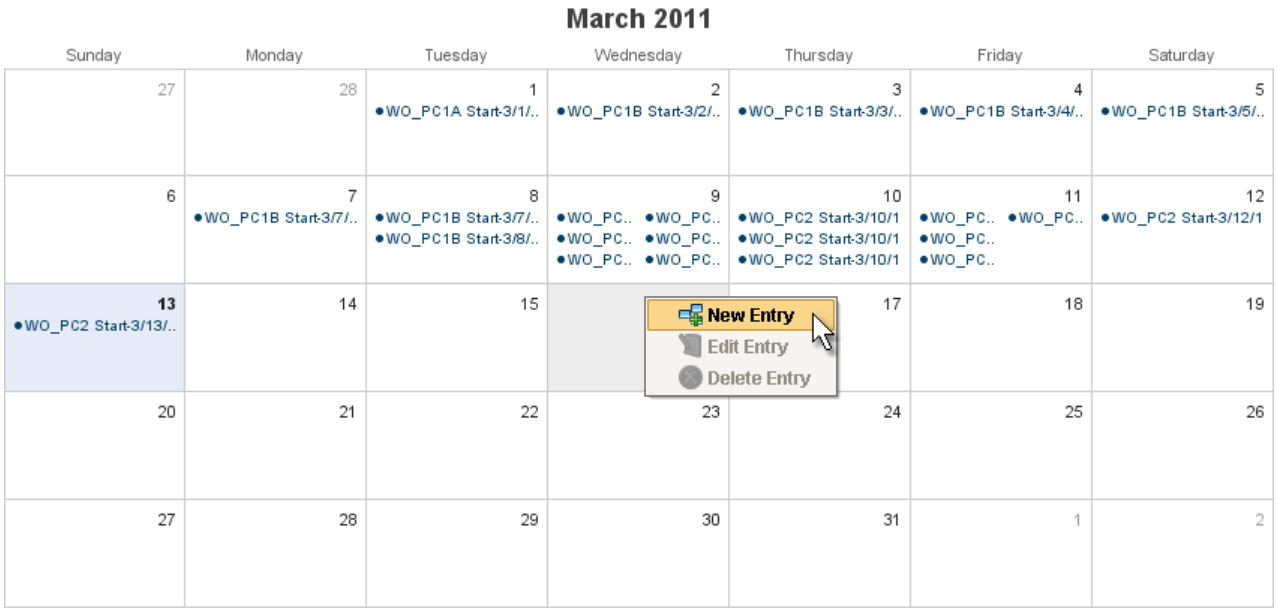
© Inductive Automation

2.4.3.7 Schedule Month View

Description

A component that displays scheduled entries for a selected month. This extends from the Month View Component that comes with Ignition to support adding, editing and deleting schedule entries. All schedule entries are automatically displayed from the "Schedule" and other database tables without the need for custom SQL statements or script.

When the user right clicks on a time, a popup menu will appear with options to add, edit or delete a schedule entry.



Schedule Month View

Properties

This component has the same properties as the Ignition Month View Component with the addition of the following properties:

Line Path The *line* path of the production *line* that this component is associated with. This is the full path name of the *line* starting with the project name.
For example: "OEEDemo\Your Enterprise\Your Site\Your Area\Line 1"

```
Scripting name     linePath
Data Type          String
```

Current Date The date to show schedule entries for. The date can be any day within the month.

```
Scripting name     currentDate
Data Type          Date
```

Schedule ID The database ID of the selected schedule event. -1 if no event is selected.

```
Scripting name     scheduleID
Data Type          Integer
```

© Inductive Automation

Event Display Format Allows a custom string with replaceable parameters to be used. Leave blank to get the default event display string.

```
Scripting name     eventDisplayFormat
Data Type          String
parameters         {wo}  = Work Order
                   {co}  = Scheduled Change Over Start Date
                   {st}  = Scheduled Production Start Date
                   {fn}  = Scheduled Finish Time Date
                   {pc}  = Product Code
                   {pcd} = Product Code description
                   {qty} = Scheduled Quantity
                   {nt}  = Note
                   {eb}  = Entered By
                   |     = line feed (the pipe character)

Example:
If you set Event Display Format property to:
       Product Code = {pc} - {pcd}|Note: {nt}
The display string would look like:
       Product Code = PC243 - 24oz, 3 pack
       Note: Short run requested by customer
```

Events

This control has the same events as the Ignition Table Component with the addition of the following events:

newEvent Is fired when the **New Entry** menu item is selected.

editEvent Is fired when the **Edit Entry** menu item is selected.

deleteEvent Is fired when the **Delete Entry** menu item is selected.

Methods

(none)

© Inductive Automation

Example Code

The following script can be entered into the newEvent event of this component. It collects the selected time when the right-click occurred and opens a new window with the collected values as parameters.

```
param1 = event.source.parent.getComponent('Production Line Selector').selectedLinePath
param2 = event.source.parent.getComponent('Schedule Week View').hoveredTimeLatched
system.nav.openWindow('ScheduleNew', {'LinePath' : param1, 'CurrentDate' : param2})
system.nav.centerWindow('ScheduleNew')
```

2.4.3.8 Schedule Date Selector

Description

A component that provides an easy method for users to select a day. It synchronizes the Schedule Day View, Schedule Week View and Schedule Month View components to all be selected to the same date.

Schedule Date Selector

Properties

This component has the same properties as the Ignition Month View Component with the addition of the following properties:

Current Date The currently selected date.

```
Scripting name     currentDate
Data Type          Date
```

Selected Day The currently selected date.

```
Scripting name     currentDate
Data Type          Strin
```

Add Day Used to adjust the currently selected day by a specified number of days forward or backwards. If the specified number of days is positive, then the current date will be adjust forward by the number of days specified. If the specified number of days is negative, then the current date will be adjust back by the number of days specified.

```
Scripting name     addDay
Data Type          Integer
```

Add Month Used to adjust the currently selected day by a specified number of months forward or backwards. If the specified number of months is positive, then the current date will be adjust forward by the number of months specified. If the specified number of months is negative, then the current date will be adjust back by the number of months specified.

```
Scripting name     addMonth
Data Type          Integer
```

Add Year Used to adjust the currently selected day by a specified number of years forward or backwards. If the specified number of years is positive, then the current date will be adjust forward by the number of years specified. If the specified number of years is negative, then the current date will be adjust back by the number of years specified.

```
Scripting name     addYear
Data Type          Integer
```

Events

This component has standard Ignition events.

Methods

(none)

Example Code

The following script can be entered into a button's actionPerformed to change the Schedule Date Selector's Current Date back 1 day.

```
event.source.parent.getComponent('Schedule Date Selector').addDay = -1
```

2.4.3.9 Schedule Entry Controller

Description

An invisible component that provides adding, editing and deleting schedule entries. The term *invisible component* means that the control appears during design time, but is not visible during runtime. Scheduled entries are stored in the "Schedule" database table and this control handles all SQL statements, duplicate checking, etc.

This component has built-in functionality to calculate finish date and time of work order type schedule entries based on the start date and time, product code, change over time, quantity and configured workday routine breaks.

Alternatively, schedule entries can added directly into the "Schedule" database table directly, bypassing the OEE Downtime and Scheduling Module. This method supports integration to ERP or other software systems.

The properties are provided so that after the **Schedule ID** property is set, selection components can be bound to them to display their current values. The methods are provided to perform adding, editing and deleting of schedule entries.

Properties

This component has standard Ignition properties with the addition of the following properties:

Schedule ID The currently selected ID of the schedule entry being edited. This is the ID for the "Schedule" database table.

```
Scripting name     scheduleID
Data Type          Integer
```

Line Path The *line* path of the production *line* that this component is associated with. This is the full path name of the *line* starting with the project name.
For example: "OEEDemo\Your Enterprise\Your Site\Your Area\Line 1"

```
Scripting name     linePath
Data Type          String
```

Work Order ID The current ID of the work order being scheduled. This is the ID for the "WorkOrder" database table.

```
Scripting name     workOrderID
Data Type          Integer
```

Work Order The work order number being scheduled.

```
Scripting name     workOrder
Data Type          String
```

Product Code The product code number associated with the work order being scheduled.

```
Scripting name     productCode
Data Type          String
```

Product Code Description The product code description associated with the work order being scheduled.

```
Scripting name     productCodeDescription
Data Type          String
```

Schedule Type The type of schedule entry.

```
Scripting name     scheduleType
Data Type          Integer
Options:
                   0 = Work Order Run
                   1 = Maintenance
                   2 = Other
```

Start Date Time The start date and time of the schedule entry.

```
Scripting name     startDateTime
Data Type          Date
```

Run Start Date Time The run start date and time of the schedule entry. The **Run Start Date Time** is the **Start Date Time** adjusted by the **Change Over Duration**. This is the date and time after change over is complete and the actual production begins.

```
Scripting name     runStartDateTime
Data Type          Date
```

© Inductive Automation

Change Over Duration The duration in minutes allowed for changeover.

```
Scripting name     changeOverDuration
Data Type          Integer
```

Finish Date Time The finish date and time for the schedule entry.

```
Scripting name     finishDateTime
Data Type          Date
```

Override the Finish Date Time If true, a manual finish date and time will be used instead of the automatic calculation to forecast the finish time.

```
Scripting name     finishDateTimeOverriden
Data Type          Boolean
```

Quantity The quantity of units to produce for this schedule entry.

```
Scripting name     quantity
Data Type          Integer
```

Note An optional note to associate with the schedule entry.

```
Scripting name     note
Data Type          String
```

Events

This component has standard Ignition events.

Methods

```
addScheduleEntry(linePath, workOrderID, scheduleType, start, coDuration,
finish, quantity, userName, note)
```

Add a new schedule entry.

parameters

`linePath` The line path of the production *line* that this component is associated with. This is the full path name of the *line* starting with the project name.
For example: "OEEDemo\Your Enterprise\Your Site\Your Area\Line 1"
Data Type String

`workOrderID` The ID of the work order to modify. This is the ID for the "WorkOr database table.
Data Type Integer

`scheduleType` The type of schedule entry.
Data Type Integer
Options:
0 = Work Order Run
1 = Maintenance
2 = Other

start	The starting date and time of the schedule entry. Data Type Date
coDuration	The duration of the changeover in minutes. Data Type Integer
finish	The ending date and time of the scheduled entry. Data Type Date
quantity	The quantity of units to produce for this schedule entry. Data Type Integer
userName	The name of the user who is adding this scheduled entry. Data Type String
note	An optional note to be tied to this scheduled entry. Data Type String

returns

message	Contains a description of any error encountered, otherwise it will empty Data Type String

---

```
editScheduleEntry(linePath, workOrderID, scheduleType, start, coDuration,
finish, quantity, userName, note, scheduleID)
```

Edit an existing schedule entry.

parameters

linePath	The line path of the production *line* that this component is associated with. This is the full path name of the *line* starting with the project name. For example: "OEEDemo\Your Enterprise\Your Site\Your Area\Line 1" Data Type String
workOrderID	The ID of the work order to modify. This is the ID for the "WorkOr database table. Data Type Integer
scheduleType	The type of schedule entry. Data Type Integer Options: 0 = Work Order Run 1 = Maintenance 2 = Other
start	The starting date and time of the schedule entry. Data Type Date
coDuration	The duration of the changeover in minutes. Data Type Integer

finish	The ending date and time of the scheduled entry. Data Type Date
quantity	The quantity of units to produce for this schedule entry. Data Type Integer
userName	The name of the user who is adding this scheduled entry. Data Type String
note	An optional note to be tied to this scheduled entry. Data Type String
scheduleID	The ID of the schedule entry to modify. This is the ID for the "Schedule" database table. Data Type Integer

returns

message	Contains a description of any error encountered, otherwise it will empty Data Type String

---

`deleteScheduleEntry(workOrderID)`

Delete an existing schedule entry.

parameters

scheduleID	The ID of the schedule entry to modify. This is the ID for "Schedule" database table. Data Type Integer

returns

message	Contains a description of any error encountered, otherw empty Data Type String

Example Code

The following script can be entered in a button's actionPerformed event. It will add the schedule entry to the database. The return message will indicate if the there are any issues adding the schedule entry. See the OEEDemo project's ScheduleNew window for a full implementation example.

```
esp = event.source.parent
# gather parameters required to add a schedule entry
linePath = esp.LinePath
workOrderID = esp.getComponent('WorkOrderContainer').getComponent('Work Order Selector').sel
scheduleType = esp.getComponent('ScheduleType').selectedValue
startDate = esp.getComponent('StartDateTime').selectedDateTime
coDuration = esp.getComponent('WorkOrderContainer').getComponent('CODuration').selectedValue
finishDate = esp.getComponent('FinishDateTime').selectedDateTime
quantity = esp.getComponent('WorkOrderContainer').getComponent('Quantity').intValue
userName = esp.getComponent('HiddenContainer').getComponent('UserName').text
note = esp.getComponent('note').text
# call the add schedule entry method of the schedule entry controller
result = esp.getComponent('Schedule Entry Controller').addScheduleEntry(linePath, workOrderI
# handle result
if (result == ''):
  esp.getComponent('WorkOrderContainer').getComponent('Work Order Selector').selectedIndex =
  system.nav.closeParentWindow(event)
```

2.4.3.10 Schedule Controller

Description

An invisible component that provides selection of scheduled entries for a specified production *line*. It also provides start, end and resume control of production runs. The term *invisible component* means that the control appears during design time, but is not visible during runtime. Scheduled entries are stored in the "Schedule" database table and this control handles all SQL statements, duplicate checking, etc.

This component has built-in functionality to calculate finish date and time of work order type of schedule entries based on the start date and time, product code, change over time, quantity and configured workday routine breaks.

Alternatively, schedule entries can added directly into the "Schedule" database table directly, bypassing the OEE Downtime and Scheduling Module. This method supports integration to ERP or other software systems.

The properties are provided so that after the **Schedule ID** property is set, selection components can be bound to them to display their current values. The methods are provided to perform adding, editing and deleting of schedule entries.

© Inductive Automation

Properties

This component has standard Ignition properties with the addition of the following properties:

Line Path
The *line* path of the production *line* that this component is associated with. This is the full path name of the *line* starting with the project name.
For example: "OEEDemo\Your Enterprise\Your Site\Your Area\Line 1"
Scripting name `linePath`
Data Type `String`

Running
The current running read only state of the production *line*. The production *line* is considered running if a production run is started and the *line* state is running.
Scripting name `running`
Data Type `Boolean`

Production Enabled
The current production enabled read only state of the production run. This will be true after the changeover time has expired or the operator initiated the production run start depending on the setting of the Auto Start property that configured in the designer.
Scripting name `productionEnabled`
Data Type `Boolean`

Can Start
This will be true if a production run can start. It is typically used to control the enable state of a "Start" button on the operator screen.
Scripting name `canStart`
Data Type `Boolean`

Started
This will be true if the production run is started.
Scripting name `started`
Data Type `Boolean`

Can End
This will be true if a production run can be ended. It is typically used to control the enable state of a "End" button on the operator screen.
Scripting name `canEnd`
Data Type `Boolean`

Can Resume
This will be true if a production run has been ended and a new schedule entry has not been selected. It is typically used to control the enable state of a "Resume" button on the operator screen.
Scripting name `canResume`
Data Type `Boolean`

Can Change Schedule
This will be true if the current schedule entry can be changed. It is typically used to control the enable state of a Line Schedule Selector component.
Scripting name `canChangeSchedule`
Data Type `Boolean`

Is Work Order
This will be true if the currently selected schedule entry is a work order type.
Scripting name `isWorkOrder`
Data Type `Boolean`

© Inductive Automation

Inhibit Start Can be set to true to prevent a production run from being started.

```
Scripting name     inhibitStart
Data Type          Boolean
```

Start Set to true to start the production run for the current Schedule ID.

```
Scripting name     start
Data Type          Boolean
```

End Set to true to stop the current production run.

```
Scripting name     end
Data Type          Boolean
```

Resume Set to true to resume the current production run.

```
Scripting name     resume
Data Type          Boolean
```

Events

This component has standard Ignition events.

Methods

`startRun()`

Start production run.

parameters (none)

returns

message	If successful, returns true.	
	Data Type	String

`endRun()`

End production run.

parameters (none)

returns

message	If successful, returns true.	
	Data Type	String

---

cancelRun()

Cancel production run.

parameters (none)

returns

message	If successful, returns true.	
	Data Type	String

---

resumeRun()

Resume production run.

parameters (none)

returns

message	If successful, returns true.	
	Data Type	String

Example Code

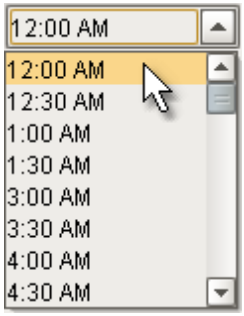
The following script can be entered in a button's actionPerformed event. It will end the current production run. See the OEEDemo project's Operator_Control window for a full implementation example.

```
value = 1
event.source.parent.getComponent('Schedule Controller').end = value
```

2.4.3.11 Time Selector

Description

A component that provides users he option to select a time from a drop-down list. The interval of time between each option is defined by the **Time Interval** property.



Time Selector

Properties

This component has standard Ignition properties with the addition of the following properties:

Time Interval The time interval between each option in the drop-down list.

```
Scripting name   linePath
Data Type        String
Options:
                 Hour
                 30 minutes
                 15 minutes
                 10 minutes
                 1 minute
```

Selected Date Time The currently selected date and time.

```
Scripting name     selectedDateTime
Data Type          Date
```

Date Part The currently selected date.

```
Scripting name     datePart
Data Type          Date
```

Selected Time The currently selected time.

```
Scripting name     selectedTime
Data Type          String
```

Events

This component has standard Ignition events.

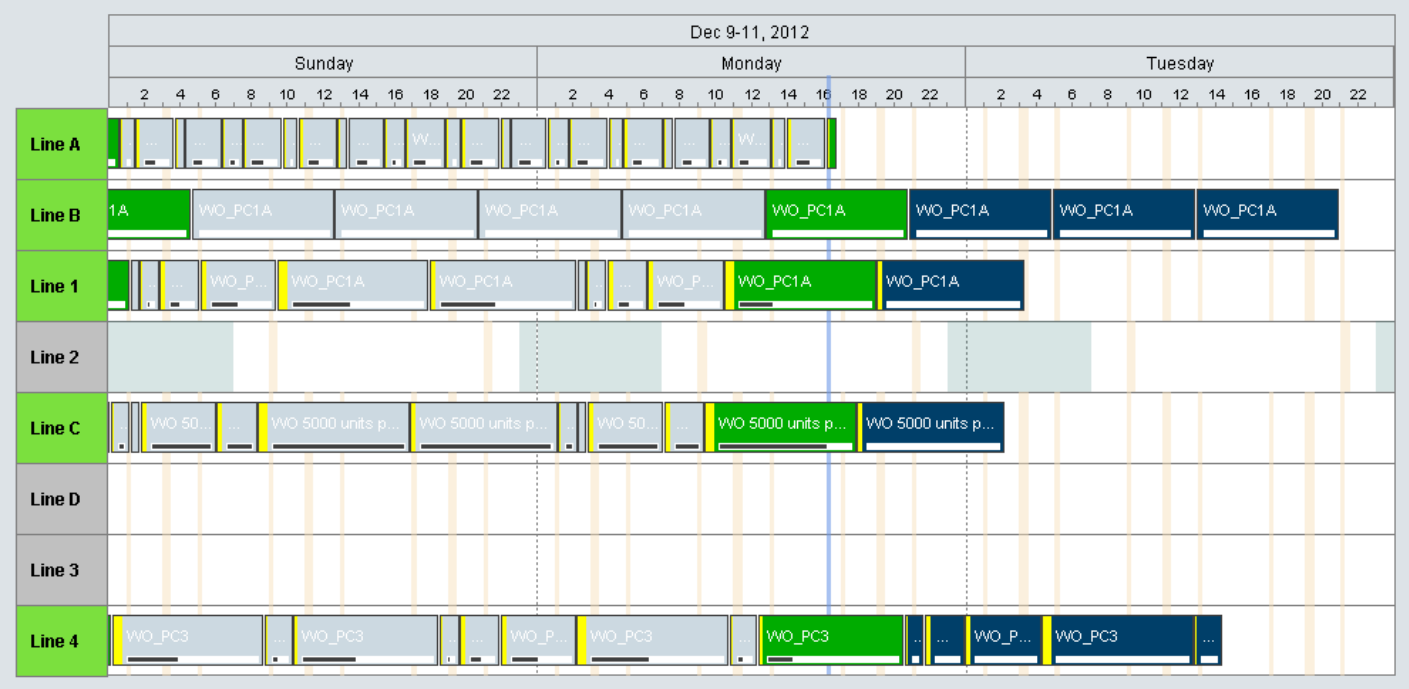
Methods

(none)

2.4.3.12 Line Schedule View

Description

A component that displays scheduled entries and status for a line, adding, editing and deleting of schedules and allows drag and drop rescheduling.



Line Schedule View

Events

This control has the common ignition events with the addition of the following events:

userMenuItemClicked

Fired when a user menu item is selected by a right mouse click.
User menu items are added with calls to the method "addUserMenuItem (<menuName>)".
The are many event properties returned that can be used to drive analysis data or other type of funtions.

Example:

```
menuItemName = event.menuItemName
if menuItemName == "Display Click Data":
        Date = event.date
        LinePath = event.linePath
        SchedId = event.scheduleID
        RunID = event.runID
        RunName = event.runName
        StartDate = event.startDate
        EndDate = event.endDate

        message = "menuItemName=%s\n Date=%s\n LinePath=%s\n SchedId=%s\n RunID=%s\n RunName=%s\n StartDate=%s\n EndDate=%s" %(menuItemName, Date, LinePath, SchedId, RunID, RunName, StartDate, EndDate)

        system.gui.messageBox(message, "User Menu Item Clicked")

if menuItemName == "Show Run Actual vs. Scheduled":
```

RunName = event.runName
system.gui.messageBox(RunName, "User Menu Item Clicked")

Event Properties

event.menuItemName — Returns the menu name of selected user menu item.

Data Type String

event.linePath — Returns the line path corresponding to where the user menu item was clicked.

Data Type String

event.date — Returns the date corresponding to where the user menu item was clicked.

Data Type Date

event.scheduleID — Returns the schedule ID of the event where the user menu item was clicked. If no event exists then it will return -1.

Data Type Integer

event.runID — Returns the run ID of the event where the user menu item was clicked. If no event exists then it will return -1.

Data Type Integer

event.runName — Returns the run name of the event where the user menu item was clicked. If no event exists then it will return blank.

Data Type String

event.startDate — Returns the start date of the event where the user menu item was clicked. If no event exists then it will return None.

Data Type Date

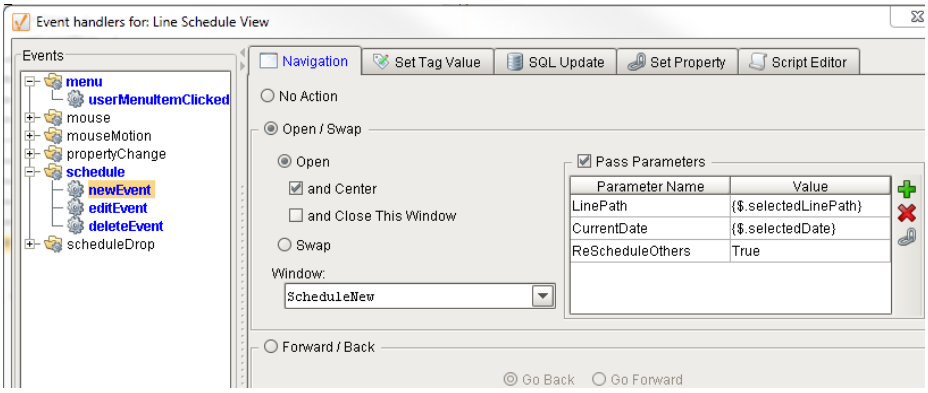
event.endDate — Returns the end date of the event where the user menu item was clicked. If no event exists then it will return None.

Data Type Date

newEvent — Fired when the new event menu item is selected by a right mouse click. Use selected component properties to create and add a new schedule.

Example:
This will open the OEE Demo applications schedule new window to add a new schedule entry. The new schedule window requires the line path and selected date.

© Inductive Automation



Implement the newEvent with OEE Demo existing window

Event Properties
(none)

editEvent

Fired when the edit event menu item is selected by a right mouse click. Use "selectedEvent" from the component to get the scheduleID of the schedule item to edit.

Event Properties
(none)

deleteEvent

Fired when the delete event menu item is selected by a right mouse click. Use "selectedEvent" from the component to get the scheduleID of the schedule item to delete.

Event Properties
(none)

Methods

`addUserMenuItem(menuName)`

Adds user menu items to the component that will display on a right click of the mouse. Use the "userMenuItemClicked" event handler to provide any functionality you define for each menu item.

Example (This script is added to the "internalFrameOpened" event for a window and will add the three user menu items.):

```
def initMenuItems(event=event):
  import system
  lineSchedView = system.gui.getParentWindow(event).getComponentForPath('Root Container.Line Schedule View')
  lineSchedView.addUserMenuItem("Display Click Data")
  lineSchedView.addUserMenuItem("Show Run Actual vs. Scheduled")
  lineSchedView.addUserMenuItem("Show Run Actual vs. Target")
```

system.util.invokeLater(initMenuItems) # allows all bindings to complete before execution

parameters

menuName	The name of the menu item. It will be returned in the userMenuItemClicked event as event.menuItemName Data Type String

returns nothing

Properties

This component has standard Ignition properties with the addition of the following properties:

Drag Enabled When true schedules can be re-scheduled with drag and drop via the mouse.

```
Scripting name     dragEnabled
Data Type          Boolean
```

Time Controls the time resolution (in minutes) when performing time operations in the component.

Example: When set to 5 (minutes) then any right click on the component will be aligned with the closest 5 minute mark or if dragging a schedule to re-schedule, it will fall into the closest 5 minute mark.

```
Scripting name     timeResolution
Data Type          Integer
```

Start Date The starting date for display.

```
Scripting name     startDate
Data Type          Date
```

End Date The ending date for display.

```
Scripting name     endDate
Data Type          Date
```

Line Filter A comma separated list of lines to display.

Example: When set to "Line A, Line B" then only Line A and Line B will be displayed.

```
Scripting name     lineFilter
Data Type          String
```

© Inductive Automation

Production Model Item Filter The production model path to filter upon.

Example: When set to "OEEDemo\Your Enterprise\Site 1\Packaging" then all lines under Packaging will be displayed.

```
Scripting name     itemPath
Data Type          String
```

Selected Event ID The selected schedule ID of any mouse click or -1 if no event item exists at the location.

```
Scripting name     selectedEvent
Data Type          Integer
```

Selected Event Date Returns the date in the view where the mouse was ***right*** clicked last.

```
Scripting name     selectedDate
Data Type          Date
```

Selected Event Line Path The selected line path of any mouse click on an event.

```
Scripting name     selectedLinePath
Data Type          String
```

Selected Event Run ID The selected Run ID of any mouse click or -1 if no event item exists at the location.

```
Scripting name     selectedRunID
Data Type          Integer
```

Selected Event Run Name The selected Run Name of any mouse click or blank if no event item exists at the location.

```
Scripting name     selectedRunName
Data Type          String
```

Selected Event Start Returns the start date of any mouse click or blank if no event item exists at the location.

```
Scripting name     selectedEventStart
Data Type          Date
```

Selected Event End Returns the end date of any mouse click or blank if no event item exists at the location.

```
Scripting name     selectedEventEnd
Data Type          Date
```

Event Border Sets the border type for non-selected event items displayed.

```
Scripting name     eventBorder
Data Type          Border
```

© Inductive Automation

Selected Event Border Sets the border type for the selected event item.

```
Scripting name     selectedEventBorder
Data Type          Border
```

Line Height Height, in pixels, of each line row.

```
Scripting name     selectedEventBorder
Data Type          Integer
```

Event Margin The margin, in pixels, from the top and bottom of a line row and the event item displayed.

```
Scripting name     scheduledEventMargin
Data Type          Integer
```

Schedule Background The color of the background of the line rows.

```
Scripting name     scheduleBackground
Data Type          Color
```

Current Time Color The color of the vertical line that indicates the current time.

```
Scripting name     nowColor
Data Type          Color
```

Line Color The color of the general dividing lines in the component.

```
Scripting name     lineColor
Data Type          Color
```

Header Font The font of the header text.

```
Scripting name     headerFont
Data Type          Font
```

Header Text Color The color of the header text.

```
Scripting name     headerTextColor
Data Type          Color
```

Header Background The color of the header background.

```
Scripting name     headerBackground
Data Type          Color
```

Progress Bar Background The color of the progress bar background. The progress bar shows the the quantity produced versus the quantity scheduled.

```
Scripting name     progressBackground
Data Type          Color
```

Progress Bar Fill The color of the progress bar fill. The progress bar shows the the quantity produced versus the quantity scheduled.

```
Scripting name     progressFill
Data Type          Color
```

© Inductive Automation

Progress Bar Border
The color of the progress bar border. The progress bar shows the the quantity produced versus the quantity scheduled.

```
Scripting name     progressBorder
Data Type          Color
```

Header Item Font
The font of the line row name header text.

```
Scripting name     itemFont
Data Type          Font
```

Event Font
The font of the text displayed in an event item.

```
Scripting name     eventFont
Data Type          Font
```

Line Item Running Color
The color of the line row header when the line is running.

```
Scripting name     lineRunningColor
Data Type          Color
```

Line Item Running Icon
The path of the icon to display in the line row header when the line is running.

```
Scripting name     lineRunningIconPath
Data Type          String
```

Line Item stopped Color
The color of the line row header when the line is stopped.

```
Scripting name     lineStoppedColor
Data Type          Color
```

Line Item stopped Icon
The path of the icon to display in the line row header when the line is stopped.

```
Scripting name     lineStoppedIconPath
Data Type          String
```

Run Completed Color
The color of a completed event item.

```
Scripting name     runCompletedColor
Data Type          Color
```

Run Running Color
The color of a running event item.

```
Scripting name     runRunningColor
Data Type          Color
```

Run Scheduled Color
The color of a scheduled event item.

```
Scripting name     runScheduledColor
Data Type          Color
```

© Inductive Automation

Run Changeover Color The color of the changeover portion of an event item.

```
Scripting name     runChangeOverColor
Data Type          Color
```

Maintenance Completed Color The color of a completed maintenance item.

```
Scripting name     maintenanceCompletedColor
Data Type          Color
```

Maintenance Running Color The color of a running maintenance item.

```
Scripting name     maintenanceRunningColor
Data Type          Color
```

Maintenance Scheduled Color The color of a scheduled maintenance item.

```
Scripting name     maintenanceScheduledColor
Data Type          Color
```

Other Completed Color The color of other completed items.

```
Scripting name     otherCompletedColor
Data Type          Color
```

Other Running Color The color of other running items.

```
Scripting name     otherRunningColor
Data Type          Color
```

Other Scheduled Color The color of other scheduled items.

```
Scripting name     otherScheduledColor
Data Type          Color
```

Break Color The color of a break time span background.

```
Scripting name     breakColor
Data Type          Color
```

Disabled Shift Color The color of a disabled shift time span background.

```
Scripting name     disabledShiftColor
Data Type          Color
```

© Inductive Automation

Event Display Format Allows a custom string with replaceable parameters to be used. Leave blank to get the default event display string.

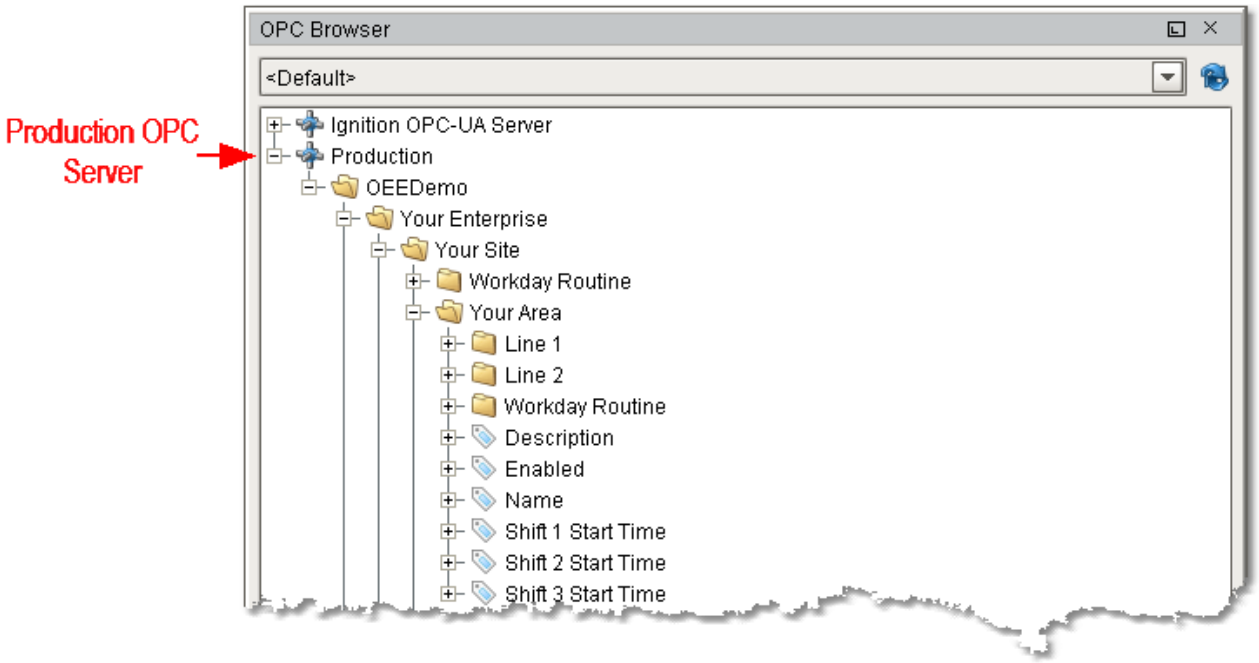
```
Scripting name      eventDisplayFormat
Data Type           String
parameters          {wo}  = Work Order
                    {co}  = Scheduled Change Over Start Date
                    {st}  = Scheduled Production Start Date
                    {fn}  = Scheduled Finish Time Date
                    {pc}  = Product Code
                    {pcd} = Product Code description
                    {qty} = Scheduled Quantity
                    {nt}  = Note
                    {eb}  = Entered By
                    |     = line feed (the pipe character)

Example:
If you set Event Display Format property to:
        Product Code = {pc} - {pcd}|Note: {nt}
The display string would look like:
        Product Code = PC243 - 24oz, 3 pack
        Note: Short run requested by customer
```

2.5 Production OPC Values

The production model is defined in the Ignition designer and contains your production *areas*, *lines* and c*ells*. A runtime access into configuration and current state of the production model is available through the Production OPC Server. It is added automatically when the OEE Downtime and Schedule Module is installed. When the production items are added, removed or modified, the changes will be reflected in the Production OPC Server when the project is saved in the designer.

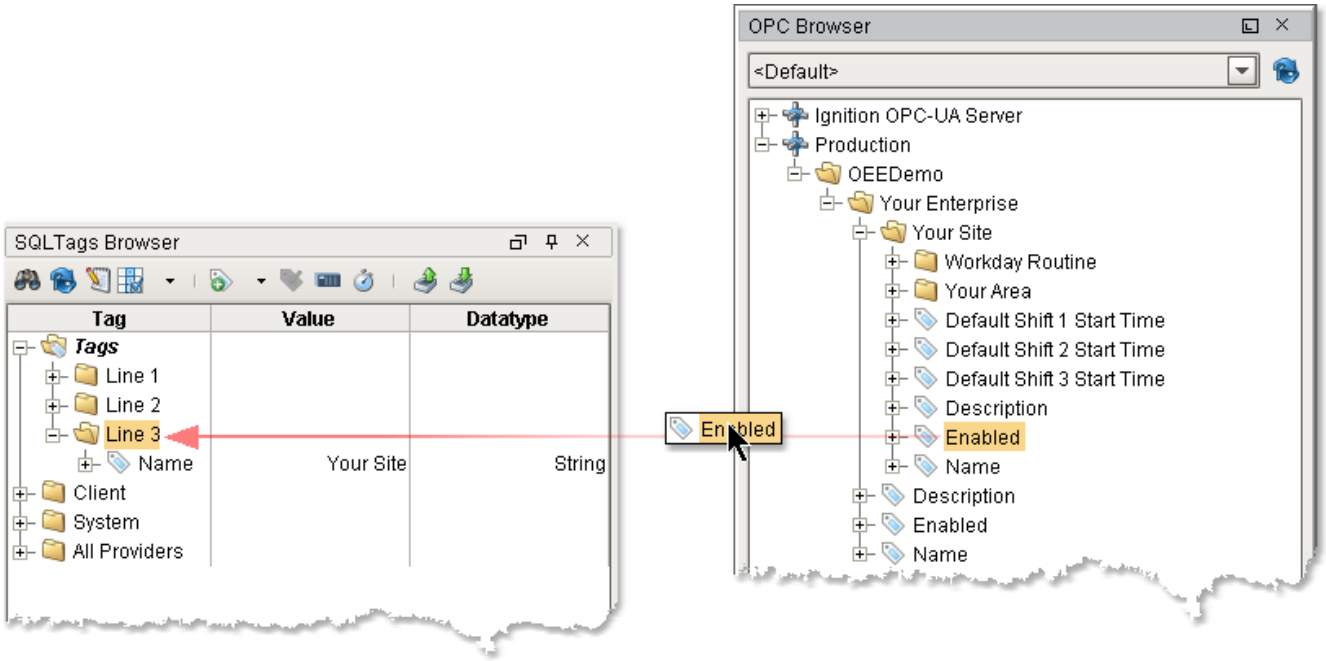
Below is a part of the values available to read, and in some cases write to, for the demo project.



Demo OPC Values

2.5.1 Using OPC Values

The OEE downtime and scheduling configuration settings and runtime values are available for use in Ignition windows, transaction groups, scripting, etc. Before values from the Production OPC Server can be used, they must be added to the Ignition SQLTags. This is done in the designer by selecting the SQLTags Browser and clicking on the icon. This will cause the OPC Browser to appear. Next, drill down in the **Production** node within the OPC Browser. Drag the desired Production OPC Values over to the SQLTags Browser as depicted below.



Add Production OPC Server Values to SQLTags

Important:
When writing to OPC values that are related to production model settings, the new value is not retained upon restarting. This is because production model settings are saved in the Ignition project and is only saved when done so in the designer.

2.5.2 OPC Value Reference

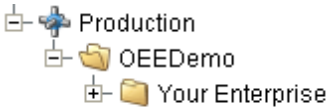
This references details the OPC values and child folders for node types that appear when browsing the Production OPC Server. For each property, the Ignition data type is listed and if it is read only. The Ignition data types correspond to the data types that are available for SQLTags.

Within this reference, the "Read Only" means that the OPC value cannot be written to through the OPC Production Server. It can only be set in the designer or it is a calculated value. Trying to write to a read only property will result in an error message being shown.

2.5.2.1 Project

Description

Each project within Ignition has its own production model. The first node(s) under the main ***Production*** node represent the Ignition project(s). Their names are the same as the project name. The image below represents the OEEDemo project.



Project

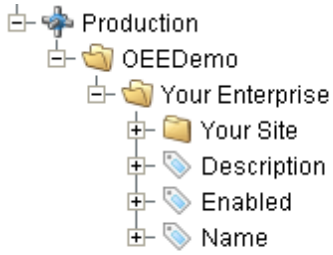
Child Folders

Enterprise One folder will exist for each *Enterprise* that has been configured in the Ignition Designer. The folder can be opened to view all values within the *enterprise*.

2.5.2.2 Enterprise

Description

The *enterprise* folder contains some properties associated with the *enterprise* and a folder for each production *Site* within it. The name is the same as the *enterprise* name that is configured in the designer. The image below represents the "Your Enterprise" of the OEEDemo project.



Enterprise

Child Folders

Site One folder will exist for each *Site* that has been configured in the Ignition Designer. The folder can be opened to view all values within the *site*.

Properties

© Inductive Automation

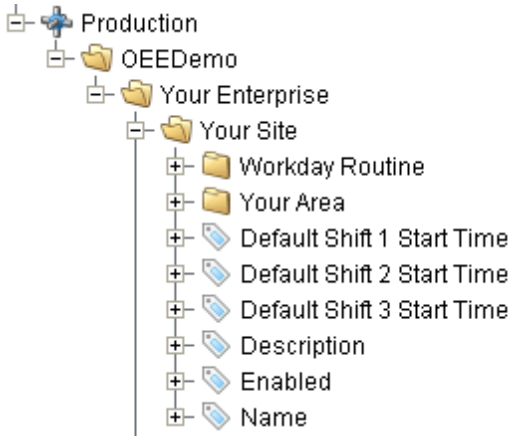
Analysis Auxiliary DB Connection Name	The name of the auxiliary (mirror) analysis database connection. Can be blank if no auxiliary DB connection is configured.	String Read Only
Analysis DB Connection Name	The name of the analysis database connection.	String Read Only
Description	Optionally, this property can be set to a description for the *enterprise*. It is not used by the OEE Downtime and Scheduling Module other than for reference.	String
Enabled	This reflects the *enterprise* Enabled property in the Designer. If the *enterprise* Enabled is set to true, then the OEE Downtime and Scheduling module will perform calculations for the *enterprise* and all *sites*, *areas*, *lines* and *cells* within it. If this property is set to false, then none of the *sites*, *areas*, *lines* or *cells* will have calculations performed.	Boolean
Name	This reflects the name of the *enterprise* that is set in the designer.	String Read Only
Runtime DB Connection Name	The name of the runtime database connection.	String Read Only
Save Control Limit by Product Code	Indicates if control limits are saved by product code. Exists only if the SPC module is also installed.	Boolean Read Only

2.5.2.3 Site

Description

The *site* folder contains some properties associated with the production *site* and a folder for each production *area* within it. The name is the same as the *site* name that is configured in the designer. The image below represents the "Your Site" of the OEEDemo project.

© Inductive Automation



Site

Child Folders

Workday Routine	Contains all of the workday routine entries that are active for the production *site*.
Area	One folder will exist for each area that has been configured in the Ignition Designer. The folder can be opened to view all values within the *area*.

Properties

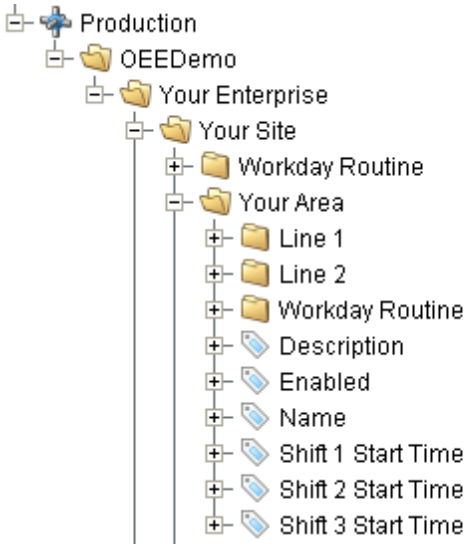
Description	Optionally, this property can be set to a description for the *site*. It is not used by the OEE Downtime and Scheduling Module other than for reference.	String
Enabled	This reflects the *site* Enabled property in the Designer. If the *site* Enabled is set to true, then the OEE Downtime and Scheduling module will perform calculations for the *site* and all *areas*, *lines* and *cells* within it. If this property is set to false, then none of the *areas*, *lines* or *cells* will have calculations performed.	Boolean
Name	This reflects the name of the *site* that is set in the designer.	String Read Only
Default Shift 1 Start Time	This reflects the *site* Default Shift 1 Start Time property in the Designer. See Site Configuration for more details.	DateTime Read Only
Default Shift 2 Start Time	This reflects the *site* Default Shift 2 Start Time property in the Designer. See Site Configuration for more details.	DateTime Read Only
Default Shift 3 Start Time	This reflects the *site* Default Shift 3 Start Time property in the Designer. See Site Configuration for more details.	DateTime Read Only

2.5.2.4 Area

Description

The *area* folder contains some properties associated with the production *area* and a folder for each production *line* within it. The name is the same as the *area* name that is configured in the designer. The image below represents the "Your Area" of the OEEDemo project.

© Inductive Automation



Area

Child Folders

Workday Routine	Contains all of the workday routine entries that are active for the production *area*.
Line	One folder will exist for each *Line* that has been configured in the Ignition Designer. The folder can be opened to view all values within the *line*.

Properties

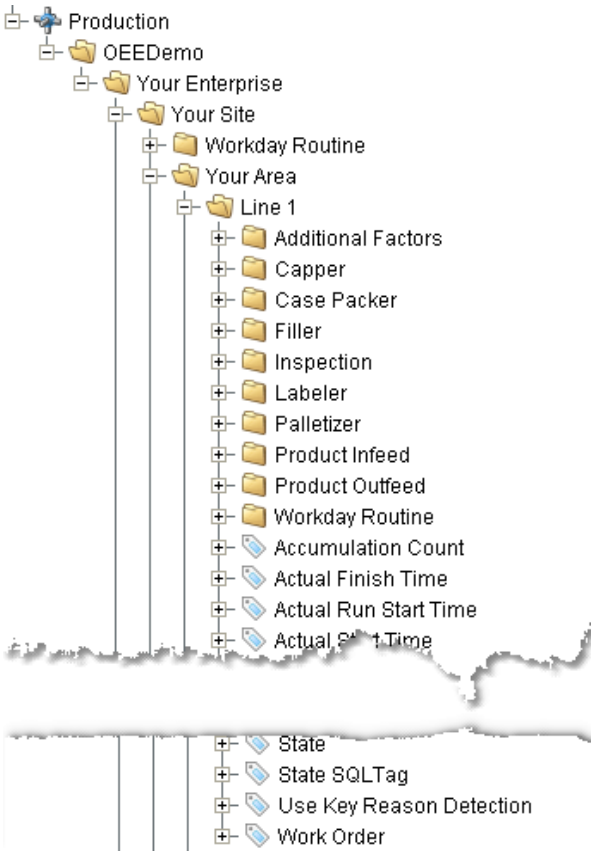
Description	Optionally, this property can be set to a description for the *area*. It is not used by the OEE Downtime and Scheduling Module other than for reference.	String
Enabled	This reflects the *site* Enabled property in the Designer. If the *area* Enabled is set to true, then the OEE Downtime and Scheduling module will perform calculations for the *area* and all *lines* and cell within it. If this property is set to false, then none of the *lines* or cells will have calculations performed.	Boolean
Name	This reflects the name of the *area* that is set in the designer.	String Read Only
Shift 1 Start Time	The current Shift 1 Start Time time for the production *area*. If the associated Shift 1 Start Time property for the *area* in the designer is set to *Inherit From Parent*, this will be the time defined for the parent production *site*. See Area Configuration for more details.	DateTime Read Only
Shift 2 Start Time	The current Shift 2 Start Time time for the production *area*. If the associated Shift 2 Start Time property for the *area* in the designer is set to *Inherit From Parent*, this will be the time defined for the parent production *site*. See Area Configuration for more details.	DateTime Read Only
Shift 3 Start Time	The current Shift 3 Start Time time for the production *area*. If the associated Shift 3 Start Time property for the *area* in the designer is set to *Inherit From Parent*, this will be the time defined for the parent production *site*. See Area Configuration for more details.	DateTime Read Only

© Inductive Automation

2.5.2.5 Line

Description

The *line* folder contains some properties associated with the production *line* and a folder for each production cell within it. The name is the same as the *line* name that is configured in the designer. The image below represents the "Line 1" of the OEEDemo project.



Line

Child Folders

Additional Factors	Contains all of the additional factor entries that have been configured for the production *line*. See Additional Factors for more details.
Downtime Reasons	Contains all of the downtime reasons entries that have been configured for the production *line*. See Downtime Reasons for more details.
Workday Routine	Contains all of the workday routine entries that are active for the production *line*. See Workday Routine for more details.
Cell	One folder will exist for each ***Cell*** that has been configured in the Ignition Designer. The folder can be opened to view all values within the cell.

© Inductive Automation

Properties

Accumulation Count	*Accumulation Count = Infeed Count - Run Production Count.* This represents the amount of product accumulated on the production *line* and is adjusted for package count. It will be the same units as the infeed.	Int4 Read Only
Active Downtime Duration	Indicates the time, in a formatted string, that the current line downtime event has been active.	String Read Only
Active Downtime Is a Short Stop	Indicates if the current active line downtime event is a short stop.	Boolean Read Only
Active Downtime Starttime	Indicates the start time that the current line downtime event started.	DateTime Read Only
Actual Changeover End Time	*I*ndicates the time that the current run ended changeover status and began running.	DateTime Read Only
Actual Finish Time	The date and time that *Enable Run* property was set to false. This typically happens when the operator clicks the **End** button.	DateTime Read Only
Actual Run Start Time	The date and time that *Enable Run* property was set to true. This typically happens when the operator clicks the **Start** button or a production run auto start occurred (See Line Configuration Schedule Settings for more details).	DateTime Read Only
Actual Start Time	The date and time that new product was selected to run on the *line*. Typically, this happens when the operator selects a new production run.	DateTime Read Only
Auto Run Schedule	Indicates if the line is set to automatically start schedule entries and begin the run after any changeover. Overrides "Auto Start Run" setting.	Boolean Read Only
Auto Start Run	Indicates if the production run should start automatically after change over. This will allow the line to record a downtime event if it is not running after the allotted changeover time from the schedule entry.	Boolean Read Only
Calculate Count	This value will increment every time OEE, downtime and scheduling values are calculated for the project production model.	Int4 Read Only
Can Cancel Run	Indicates if this run can be cancelled. Runs can only be cancelled while in changeover	Boolean Read Only
Can Resume Run	If true, all conditions are good to resume a production run.	Boolean Read Only

© Inductive Automation

Can Start Run	If true, all conditions are good to start a production run.	Boolean Read Only
Changeover Overrun Reason Code	Reference of the downtime reason code that will be triggered on a changeover overrun.	Int4 Read Only
Description	Optionally, this property can be set to a description for the *line*. It is not used by the OEE Downtime and Scheduling Module other than for reference.	String
Downtime Detection Method	This reflects the current value of the "Downtime Detection Method" setting in the designer.	String Read Only
Downtime Reason Description	Indicates the reason description of the current line downtime active event.	String Read Only
Enable Run	Setting *Enable Run* to true will enable the production run for the *line*. Setting it to false will end the production run. Typically, this is controlled by the functionality of the operator screen, but it can also be handled programmatically.	Boolean
Enabled	This reflects the *line* Enabled property in the Designer. If the *line* Enabled is set to true, then the OEE Downtime and Scheduling module will perform calculations for the all cells within it. If this property is set to false, then none of the cells will have calculations performed.	Boolean
Infeed Count	The true unit count at the primary product infeed for the production run. The true unit count reflects the start of production run count and raw count rollovers.	Int4 Read Only
Name	This reflects the name of the *line* that is set in the designer.	String Read Only
OEE	The current OEE value for the current shift. See OEE for more details.	Float8 Read Only
OEE Availability	The current OEE Availability value for the current shift. See OEE for more details.	Float8 Read Only
OEE Performance	The current OEE Performance value for the current shift. See OEE for more details.	Float8 Read Only
OEE Quality	The current OEE Quality value for the current shift. See OEE for more details.	Float8 Read Only
Prerun Remaining Time	This is the amount of change over time, in minutes, remaining before the scheduled run start time .	String Read Only

© Inductive Automation

Prerun Remaining Time (Seconds)	This is the amount of change over time, in seconds, remaining before the scheduled run start time .	Int4 Read Only
Product Code	The current product code being run on the *line*. Typically, this is controlled by the functionality of the operator screen, but it can also be handled programmatically. It should only be changed when *Enable Run* is false.	String
Product Code Description	The description for the current *Product Code.*	String Read Only
Production Package Count	The current package count of the primary outfeed.	Int4 Read Only
Production Rate (Hour)	The current hourly production rate of the primary product outfeed. See Production Rate Calculation for more details.	Float8 Read Only
Production Rate (Minute)	The current production rate per minute of the primary product outfeed. See Production Rate Calculation for more details.	Float8 Read Only
Production Units	The units of the production rate. This reflects the units defined in the primary product outfeed. See Product Outfeed for more details.	String Read Only
Run Disabled Reason Code	This reflects the value of the "Run Disabled Reason Code" setting in the designer.	Int4 Read Only
Run Down Time (Minutes)	The total amount of unplanned downtime, in minutes, for the current production run.	Float8 Read Only
Run Elapsed Time (Minutes)	The total minutes that have elapsed from the start of the production run.	Float8 Read Only
Run ID	This is the unique identification number that was generated by the database when a row is inserted into the Run table. It can be used to associate external data to a production run.	Int4 Read Only
Run Ideal Standard Count	The ideal production count, to the minute, for the current production run based on the standard rate. This is based on the time the *line* is scheduled to run.	Int4 Read Only
Run Planned Down Time (Minutes)	The total amount of planned downtime, in minutes, for the current production run.	Float8 Read Only
Run Production Count	The total production count that has been produced for the current production run. It is in the primary product outfeed units.	Int4 Read Only

© Inductive Automation

Run Standard Count	The ideal production count, to the minute, for the current production run based on the standard rate. This is based on the time the *line* has been running, not counting any downtime.	Int4 Read Only
Run Standard Variance	The variance between the *Run Standard Count* and the *Run Production Count*.	Int4 Read Only
Run Start Date Time	This will equal the time that the production run started or the beginning of the current shift, whichever occurred last.	DateTime Read Only
Run Started	The value will be true if a production run has started. Even if the production run has been ended but a new production run has not been selected, this value will be true.	Boolean Read Only
Run Target Count	The ideal production count, to the minute, for the current production run based on the scheduling rate.	Int4 Read Only
Run Target Variance	The variance between the *Run Target Count* and the *Run Production Count*.	Int4 Read Only
Run Time (Minutes)	The total minutes that the production *line* has run for the current production run. This value excludes planned and unplanned downtime.	Float8 Read Only
Run Waste Count	See Product Waste for more details on how this value is calculated.	Int4
Running	This value will be true if a production run is started and production *line* is running.	Boolean Read Only
Schedule Rate	The current schedule rate based on the selected product code and *line*.	Float8 Read Only
Schedule Rate Period	The period of time used for the scheduling rate. The options are Hour and Minute.	String Read Only
Scheduled Finish Time	The production run finish date and time as it appears on the schedule.	DateTime Read Only
Scheduled Quantity	The total quantity to produce as it appears on the schedule.	Int4 Read Only
Scheduled Run Start Time	The start date and time of the production run as it appears on the schedule.	DateTime Read Only
Scheduled Start Time	The start date and time of the change over as it appears on the schedule.	DateTime Read Only
Sequence No	A number that is 0 at the beginning of a production run and increments at the beginning of every shift.	Int4 Read Only

© Inductive Automation

Shift	The current shift based on the shift start times configured for the production *line*.	Int4 Read Only
Shift 1 Enabled	The current Shift 1 enabled state for the production *line*. It reflects the Shift 1 Enabled property for the *line* in the designer. The initial value of this property is determined by the Shift 1 Initial Enabled State property for the production *line* in the designer. See Line Configuration for more details. It can be changed from the initial value.	Boolean
Shift 1 Start Time	The current Shift 1 Start Time time for the production *line*. If the associated Shift 1 Start Time property for the *line* in the designer is set to *Inherit From Parent*, this be the time defined for the parent production *area*. See Line Configuration for more details.	DateTime Read Only
Shift 2 Enabled	The current Shift 2 enabled state for the production *line*. It reflects the Shift 2 Enabled property for the *line* in the designer. The initial value of this property is determined by the Shift 2 Initial Enabled State property for the production *line* in the designer. See Line Configuration for more details. It can be changed from the initial value.	Boolean
Shift 2 Start Time	The current Shift 2 Start Time time for the production *line*. If the associated Shift 2 Start Time property for the *line* in the designer is set to *Inherit From Parent*, this be the time defined for the parent production *area*. See Line Configuration for more details.	DateTime Read Only
Shift 3 Enabled	The current Shift 3 enabled state for the production *line*. It reflects the Shift 3 Enabled property for the *line* in the designer. The initial value of this property is determined by the Shift 3 Initial Enabled State property for the production *line* in the designer. See Line Configuration for more details. It can be changed from the initial value.	Boolean
Shift 3 Start Time	The current Shift 3 Start Time time for the production *line*. If the associated Shift 3 Start Time property for the *line* in the designer is set to *Inherit From Parent*, this be the time defined for the parent production *area*. See Line Configuration for more details.	DateTime Read Only

© Inductive Automation

Shift Down Time (Minutes)	The total minutes of unplanned downtime for the current shift.	Float8 Read Only
Shift Elapsed Time (Minutes)	The total minutes that have elapsed from the start of the shift.	Float8 Read Only
Shift Infeed Count	The true unit count at the primary product infeed for the current shift. The true unit count reflects the start of shift count and raw count rollovers.	Int4 Read Only
Shift Production Count	The total production count that has been produced for the current shift. It is in the primary product outfeed units.	Int4 Read Only
Shift Run Time (Minutes)	The total minutes that the production *line* has run for the current shift. This value excludes planned and unplanned downtime.	Float8 Read Only
Shift Scheduled Count	The total number of units that should be produced for the current shift. If a production run extends over multiple shifts, this value is calculated for the current shift. This value is adjusted for previous shift true production whether it did not achieve or exceeded its target.	Int4 Read Only
Shift Scheduled Finish Time	This value will equal whichever is less of the forecasted production run completion time and the end of the current shift.	DateTime Read Only
Shift Standard Count	The ideal production count, to the minute, for the current shift based on the standard rate.	Int4 Read Only
Shift Standard Variance	The variance between the *Shift Standard Count* and the *Shift Production Count*.	Int4 Read Only
Shift Target Count	The ideal production count, to the minute, for the current shift based on the scheduling rate.	Int4 Read Only
Shift Target Variance	The variance between the *Shift Target Count* and the *Shift Production Count*.	Int4 Read Only
Shift Waste Count	The amount that the *Run Waste Count* increased for the current shift.	Int4 Read Only
Standard Rate	The current standard rate based on the selected product code and *line*.	Float8 Read Only
Standard Rate Period	The period of time used for the standard rate. The options are Hour and Minute.	String Read Only
State SQL Tag	This reflects the State SQLTag setting that the production *line* is configured for in the designer. It is the name of the SQLTag to read the current production *line* state from.	String Read Only

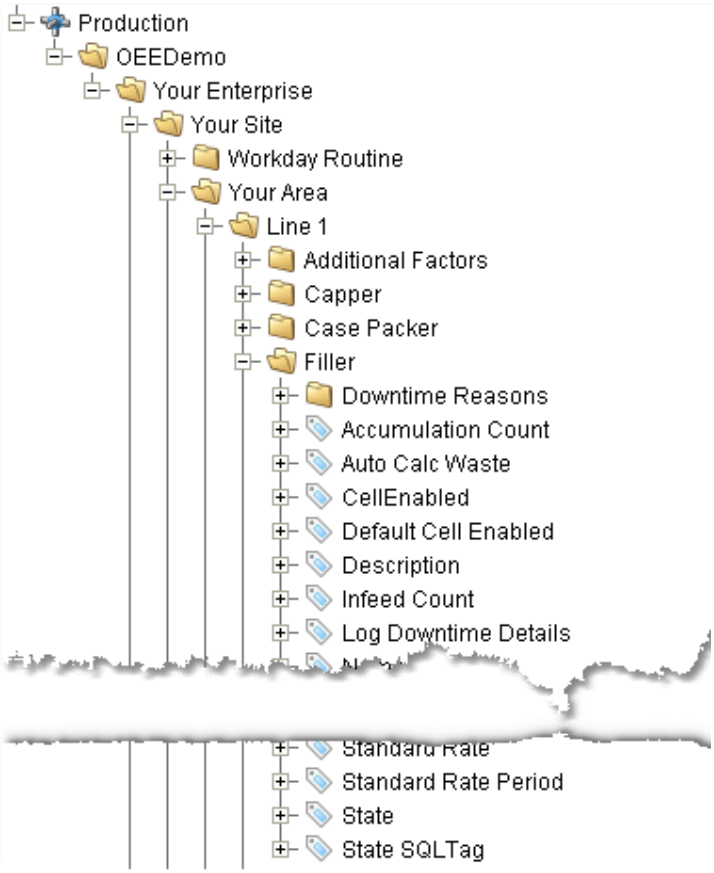
© Inductive Automation

Work Order	The current work order number for the current production run.	String Read Only

2.5.2.6 Cell

Description

The cell folder contains some properties associated with the production cell. The name is the same as the cell name that is configured in the designer. The image below represents the **Filler** of the OEEDemo project.



Cell

Child Folders

Downtime Reasons	Contains all of the downtime reasons entries that have been configured for the production cell. See Downtime Reasons for more details.

Properties

Accumulation Count	*Accumulation Count* = *Infeed Count* - *Run Production Count.* This represents the amount of product accumulated on the production *line* and is adjusted for package count. It will be the same units as the infeed.	Int4 Read Only
Active	Indicates the time, in a formatted string, that the current line downtime event has been active.	String Read Only
Duration Active	Indicates if the current active line downtime event is a short stop.	Boolean Read Only
Is a Short Stop Active	Indicates the start time that the current line downtime event started.	DateTime Read Only
Starttime		
Cell Enabled	If Cell Enabled is set to true, then the OEE Downtime and Scheduling module will perform calculations for the cell. This value is determined by the product code and production *line*. It can also be programmatically changed.	Boolean Read Only
Default Cell Enabled	This reflects the *site* Default Cell Enabled property in the Designer.	Boolean Read Only
Description	Optionally, this property can be set to a description for the cell. It is not used by the OEE Downtime and Scheduling Module other than for reference.	String
Downtime	The description of the currently active downtime for the cell.	String Read Only

© Inductive Automation

Description		
Infeed Count	The true unit count at the primary product infeed for the production run. The true unit count reflects the start of production run count and raw count rollovers.	Int4 Read Only
Line DT Bypass	When true (checked) indicates the cell will not be considered during line downtime determination. This tag can be set at runtime.	Boolean
Line DT Bypass Reason Code	Displays the selected reason code that will be used when the cell is bypassed.	Int4 Read Only
Log Downtime	Indicates if the cell will log all downtime events. If false then you will not be able to see any non-line affecting events.	Boolean Read Only
Name	This reflects the name of the cell that is set in the designer.	String Read Only
OEE	The current OEE value for the current shift. See OEE for more details.	Float8 Read Only
OEE Availability	The current OEE Availability value for the current shift. See OEE for more details.	Float8 Read Only
OEE Performance	The current OEE Performance value for the current shift. See OEE for more details.	Float8 Read Only
OEE Quality	The current OEE Quality value for the current shift. See OEE for more details.	Float8 Read Only
Production Package Count	The current package count of the primary outfeed.	Int4 Read Only
Production Rate (Hour)	The current hourly production rate of the primary product outfeed. See Production Rate Calculation for more details.	Float8 Read Only
Production Rate (Minute)	The current production rate per minute of the primary product outfeed. See Production Rate Calculation for more details.	Float8 Read Only
Production Units	The units of the production rate. This reflects the units defined in the primary product outfeed. See Product Outfeed for more details.	String Read Only
Run Down Time (Minutes)	The total amount of unplanned downtime, in minutes, for the current production run.	Float8 Read Only

© Inductive Automation

Run ID	This is the unique identification number that was generated by the database when a row is inserted into the Run table. It can be used to associate external data to a production run.	Int4 Read Only
Run Ideal Standard Count	The ideal production count, to the minute, for the current production run based on the standard rate. This is based on the time the *line* is scheduled to run.	Int4 Read Only
Run Planned Down Time (Minutes)	The total amount of planned downtime, in minutes, for the current production run.	Float8 Read Only
Run Production Count	The total production count that has been produced for the current production run. It is in the primary product outfeed units.	Int4 Read Only
Run Standard Count	The ideal production count, to the minute, for the current production run based on the standard rate.	Int4 Read Only
Run Standard Variance	The variance between the *Run Standard Count* and the *Run Production Count*.	Int4 Read Only
Run Target Count	The ideal production count, to the minute, for the current production run based on the scheduling rate.	Int4 Read Only
Run Target Variance	The variance between the *Run Target Count* and the *Run Production Count*.	Int4 Read Only
Run Time (Minutes)	The total minutes that the production *line* has run for the current production run. This value excludes planned and unplanned downtime.	Float8 Read Only
Run Waste Count	See Product Waste for more details on how this value is calculated.	Int4
Running	This value will be true if a production run is started and production *line* is running.	Boolean Read Only
Sequence No	A number that is 0 at the beginning of a production run and increments at the beginning of every shift.	Int4 Read Only
Shift Down Time (Minutes)	The total minutes of unplanned downtime for the current shift.	Float8 Read Only
Shift Elapsed Time (Minutes)	The total minutes that have elapsed from the start of the shift.	Float8 Read Only
Shift Infeed Count	The true unit count at the primary product infeed for the current shift. The true unit count reflects the start of shift count and raw count rollovers.	Int4 Read Only
Shift Production Count	The total production count that has been produced for the current shift. It is in the primary product outfeed units.	Int4 Read Only
Shift Run Time (Minutes)	The total minutes that the production *line* has run for the current shift. This value excludes planned and unplanned downtime.	Float8 Read Only

© Inductive Automation

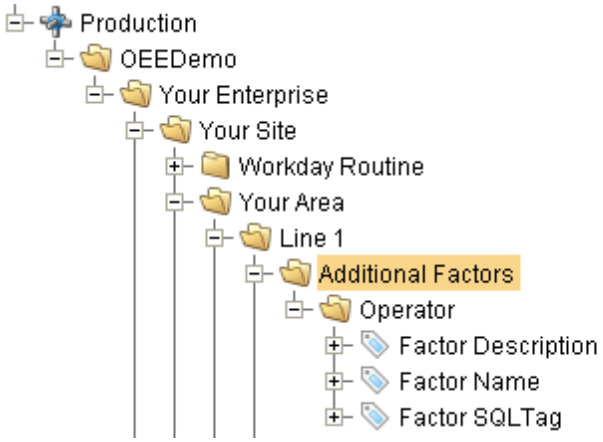
Shift Scheduled Count	The total number of units that should be produced for the current shift. If a production run extends over multiple shifts, this value is calculated for the current shift. This value is adjusted for previous shift true production whether it did not achieve or exceeded its target.	Int4 Read Only
Shift Scheduled Finish Time	This value will equal whichever is less of the forecasted production run completion time and the end of the current shift.	DateTime Read Only
Shift Standard Count	The ideal production count, to the minute, for the current shift based on the standard rate.	Int4 Read Only
Shift Standard Variance	The variance between the *Shift Standard Count* and the *Shift Production Count*.	Int4 Read Only
Shift Target Count	The ideal production count, to the minute, for the current shift based on the scheduling rate.	Int4 Read Only
Shift Target Variance	The variance between the *Shift Target Count* and the *Shift Production Count*.	Int4 Read Only
Shift Waste Count	The amount that the *Run Waste Count* increased for the current shift.	Int4 Read Only
Standard Rate	The current standard rate based on the selected product code and *line*.	Float8 Read Only
Standard Rate Period	The period of time used for the standard rate. The options are Hour and Minute.	String Read Only
State	The current state for the production *line*. The value of 0 is reserved for idle or *line* powered off and 1 is reserved for running. All other values are defined in the downtime reasons for the production *line*. See Line Configuration for more details.	Int4
State SQL Tag	This reflects the State SQLTag setting that the production *line* is configured for in the designer. It is the name of the SQLTag to read the current production *line* state from.	String Read Only

2.5.2.7 Additional Factors

Description

The additional factors folder contains a folder for each additional factor within it. The name of each folder is the same as the additional factor name that is configured in the designer. The image below represents the "Line 1" additional factors of the OEEDemo project. In the OEEDemo there is one additional factor to track the operator during a production run. See Line Configuration and Additional Factors for more details.

© Inductive Automation



Additional Factors

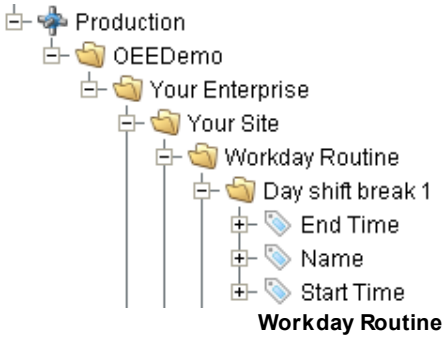
Properties

Factor Description	Optionally, this property can be set to a description for the additional factor. It is not used by the OEE Downtime and Scheduling Module other than for reference.	String
Factor Name	This reflects the name of the additional factor that is configured in the designer.	String Read Only
Factor SQLTag	This reflects the Factor SQLTag setting that the additional factor is configured for in the designer. It is the name of the SQLTag to read the factor value from.	String Read Only

2.5.2.8 Workday Routine

Description

The workday routine folder contains a folder for each workday routine entry within it. The name of each folder is the same as the workday routine entry name that is configured in the designer. The image below represents the *Site* workday routine entries of the OEEDemo project. See Workday Routines for more details.



Workday Routine

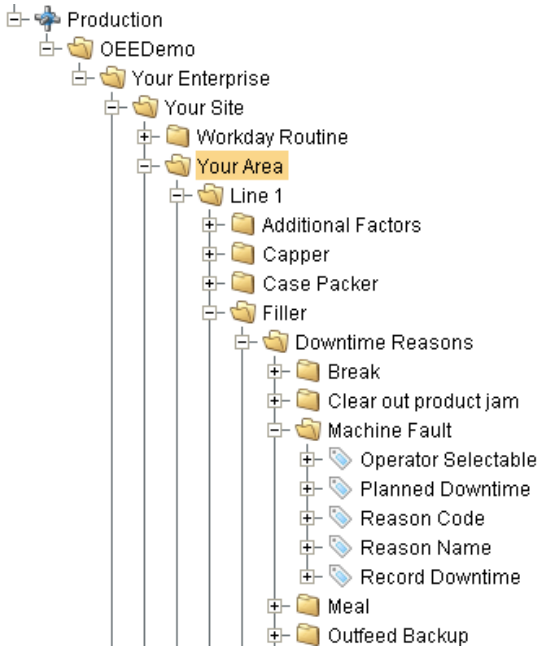
Properties

Name	This reflects the name of the workday routine entry that is configured in the designer.	String Read Only
Start Time	This reflects the *Start Time* setting that the workday routine entry is configured for in the designer. It is the time that the workday routine starts.	DateTime Read Only
End Time	This reflects the *End Time* setting that the workday routine entry is configured for in the designer. It is the time that the workday routine ends.	DateTime Read Only

2.5.2.9 Downtime Reasons

Description

The downtime reason folder contains a folder for each downtime reason entry within it. The name of each folder is the same as the downtime reason entry name that is configured in the designer. The image below represents the ***Filler*** cell downtime reason entries of the OEEDemo project. See Downtime Reasons for more details.



Downtime Reasons

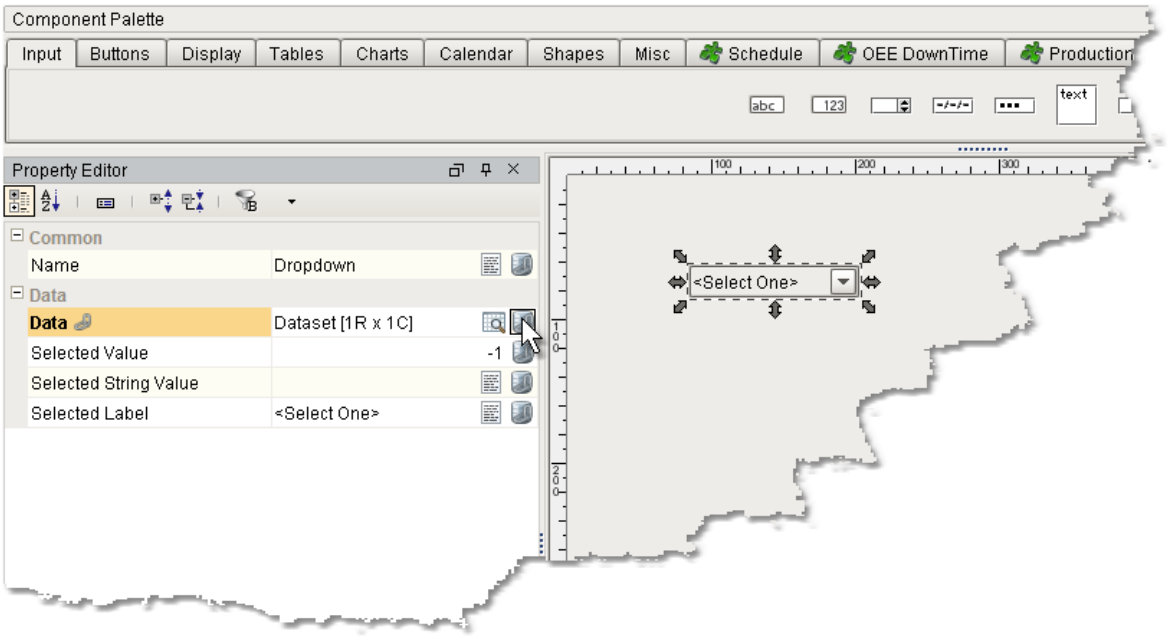
Properties

Reason Name	This reflects the *Reason Name* property of the downtime reason entry that is configured in the designer.	String Read Only
Reason Code	This reflects the *Reason Code* property of the downtime reason entry that is configured in the designer.	Int4 Read Only
Record Downtime	This reflects the *Record Downtime* property of the downtime reason entry that is configured in the designer. If true, downtime events with this reason code will count as unplanned downtime during the OEE calculation.	Boolean Read Only
Planned Downtime	This reflects the *Planned Downtime* property of the downtime reason entry that is configured in the designer. If true, downtime events with this reason code will count as planned downtime during the OEE calculation.	Boolean Read Only
Operator Selectable	This reflects the *Operator Selectable* property of the downtime reason entry that is configured in the designer. If true, the downtime reason will be shown in the Down Time Table. See Down Time Table for more details.	Boolean Read Only

2.6 Binding Function Reference

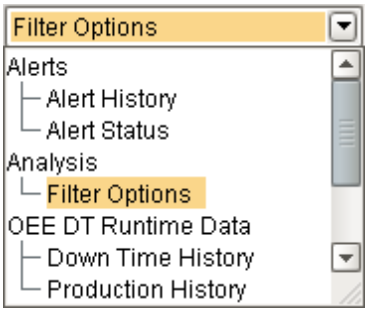
The OEE Downtime and Scheduling Module takes advantage of Ignition's built-in binding functions in order to provide data to the standard components within Ignition.

To access the binding functions, click on the icon of a component property as shown below.



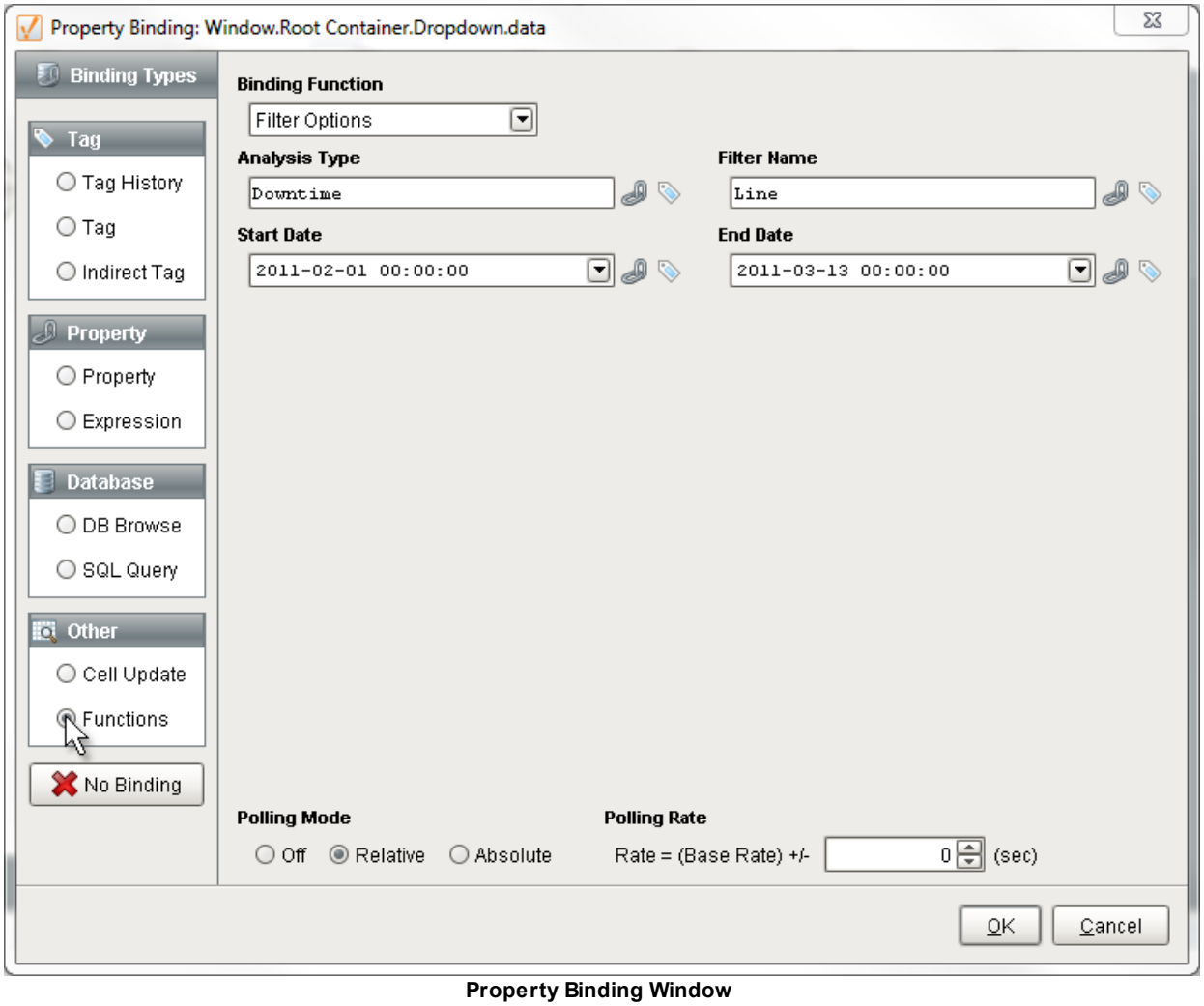
Drop-Down List Component

The binding options window will appear. Next click on the **Functions** option and select one of the binding functions from the drop-down list.



Binding Options List

The parameters that are associated with the selected binding function will appear. Each of these parameters can accept a constant value, bound to a property of another component, or bound to a SQLTag.



Property Binding Window

Once the parameters have been set and the polling mode selected, the server will return the results based on the provided parameter values.

2.6.1 Analysis

The following binding functions are provided by the Production Module, which comes with the OEE Downtime and Scheduling Module.

2.6.1.1 Analysis Filter

Description
The Analysis Filter binding function is used to return available filter values for the Analysis Controller Component . Normally this is automatically handled by the Analysis Selector Component , but for the Analysis Controller, these filter values are not known. This binding function can provide filter option data to a drop-down list or other types of components.

© Inductive Automation

Function Name
Filter Options

Parameters

Analysis Type	This parameter is the provider name that will be used. See Analysis Providers for available options.	String
Filter Name	This parameter is the name of the filter for which available options will be returned. See Analysis Providers for available options.	String
Start Date	The starting date range. To reduce the number of options, only the options for the selected date range will be returned.	Date
End Date	The ending date range. To reduce the number of options, only the options for the selected date range will be returned.	Date

Return

Filter Options	This binding function returns a Dataset with one string column with the available filter options.	Dataset

2.6.2 History

The following binding functions are provided by the OEE Downtime Module, which comes with the OEE Downtime and Scheduling Module.

2.6.2.1 Downtime History

Description
The Down Time History binding function is used to return historical downtime data for a production run. This data is gathered from the runtime database tables. This binding function can provide downtime data to tables, charts or other types of components.

If the current run is selected, downtime data from the current production run will be returned.

Function Name
Down Time History

Parameters

Production Line or Cell Path	The line or cell path of the production item that this component is associated with. This is the full path name of the *line* or cell starting with the project name. If the path ends with a *line*, the *line* downtime will be returned. If the path includes a cell, then downtime for the specified cell will be returned. For example: "OEEDemo\Your Enterprise\Your Site\Your Area\Line 1".	String
Include Total Downtime	If this parameter value is true, then total downtime for the production *line* will be included in the results	Boolean

Run ID	The production run ID for which data will be returned. This is the ID for the "Run" database table. If this parameter is set to -1 or left blank, data for the current production run for the specified production *line* will be returned.	Integer
Run Sequence No	The sequence number starts at 0 when a production run starts. It is incremented by one at the start of a new shift. This provides a method to limit results for a single shift or production runs that span over multiple days.	Integer
Include Entire Run	If this parameter value is true, all shifts for the production run are returned, If it is false, then only the shift specified by the value in **Run Sequence No** parameter will be returned.	Boolean
Top Reasons to Show	The number of top downtime reasons to return is determined by the value of this parameter.	Integer

Return

Downtime History	This binding function returns a Dataset with a variable number of columns based in the parameter settings.	Dataset

2.6.2.2 Production History

Description

The Production History binding function is used to return historical runtime data for a production run. The data for this binding function is gathered from the runtime database tables. The Production History binding function can provide production run data to tables, charts or other types of components.

If the current run is selected, production data from the current production run will be returned.

Function Name

Production History

Parameters

Production Line Path	The *line* path of the production *line* that this component is associated with. This is the full path name of the *line* starting with the project name. For example: "OEEDemo\Your Enterprise\Your Site\Your Area\Line 1".	String
Run ID	The production run ID for which data will be returned. This is the ID for the "Run" database table. If this parameter is left blank or set to -1, data for the current production run for the specified production *line* will be returned.	Integer
Run Sequence No	The sequence number starts at 0 when a production run starts. It is incremented by one at the start of a new shift. This provides a method to limit results for a single shift or production runs that span over multiple days.	Integer

Include Entire Run	If this parameter value is true, all shifts for the production run are returned, If it is false, then only the shift specified by the value in **Run Sequence No** parameter will be returned.	Boolean
Interval	This parameter specifies the time interval that the results are to be to organized by.	Hour, Minute
Number Minute to Show	This parameter specifies the minimum number of minutes in which the results should be returned. This keeps chart appearance from shifting on each update.	Integer
Include Actual Production Counts	If this parameter value is true, actual production counts will be included in the results.	Boolean
Include Standard Production Counts	If this parameter value is true, standard production counts will be included in the results. Standard production counts are based on the standard rate.	Boolean
Include Target Production Counts	If this parameter value is true, target production counts will be included in the results. Target production counts are based on the scheduling rate.	Boolean
Include Line Accumulation Counts	If this parameter value is true, *line* accumulation counts will be included in the results. Accumulation counts reflect the difference of the infeed and outfeed counts. In other terms, the amount of product that has accumulated on the production *line*.	Boolean
Include Efficiency Values	If this parameter value is true, the percentage of efficiency will be included in the results.	Boolean

Return

Production History	This binding function returns a Dataset with a variable number of columns based in the parameter settings.	Dataset

2.6.2.3 Scheduled vs. Actual

Description

The Scheduled vs. Actual binding function is used to return scheduled and actual data for the selected production run. Primarily used as the Series Data for the Analysis Time Chart component.

Function Name

Scheduled vs. Actual

Parameters

Production Line Path	The *line* path of the production *line*. This is the full path name of the *line* starting with the project name. This parameter is commonly bound to a Production Line Selector component. For example: "OEEDemo\Your Enterprise\Your Site\Your Area\Line 1".	String

Run ID	The production run ID for which data will be returned. This is the ID for the "Run" database table. If this parameter is left blank or set to -1, data for the current production run for the specified production *line* will be returned. This parameter is commonly bound to Line Run Selector component.	Integer
Sequence No	The sequence number starts at 0 when a production run starts. It is incremented by one at the start of a new shift. This provides a method to limit results for a single shift or production runs that span over multiple days. May be left blank to display all shifts.	Integer
Normalize Start Times	If this parameter value is true, the actual start will be offset to be equal to the scheduled start time. This is usefull for displaying run that were not started close to the scheduled time and allows comparison of scheduled and actual to be aligned.	Boolean
Include Scheduled	If this parameter value is false the scheduled data will not be included.	Boolean
Include Actual	If this parameter value is false the actual run data will not be included.	Boolean
Include Cells	If this parameter value is true, actual cell run data for each cell will be included in the results.	Boolean
Include Other Downtime	If this parameter value is true, downtime events that are neither planned or unplanned will be returned in the results. For instance, a cell may have a downtime event for outfeed backup that is not set to be a recordable or a planned downtime.	Boolean
Include User Comments	If this parameter value is true, the results will contain reference to any user comments entered during the actual run. The Analysis Time Chart component can display the reference.	Boolean
Stop Type	Determines the type of downtime stops to return. Both will return Long and Short stops. See the Short Downtime Threshold seconds section for more information on stop types. `Values`: Both, Short Stops, Long Stops	String

Return

Scheduled vs. Actual	This binding function returns a Dataset with a variable number of columns based in the parameter settings.	Dataset

2.7 Scripting

The OEE Downtime modules support various scripting functions that are availably on both the client and the gateway. The Line Event scripts are by line and cell and allow performing custom tasks when the line execution cycle is occurring. The custom OEE calculation scripts allow changing the default OEE calculations. When overriding the default scripts, it will be effective for run and shift OEE calculation as well as historical analysis. The Ignition gateway scripts will run on the gateway so that clients do not have to be running for them to be used. And last there are many events, methods and properties associated with the components. These are not covered in this section. See Component Reference for more

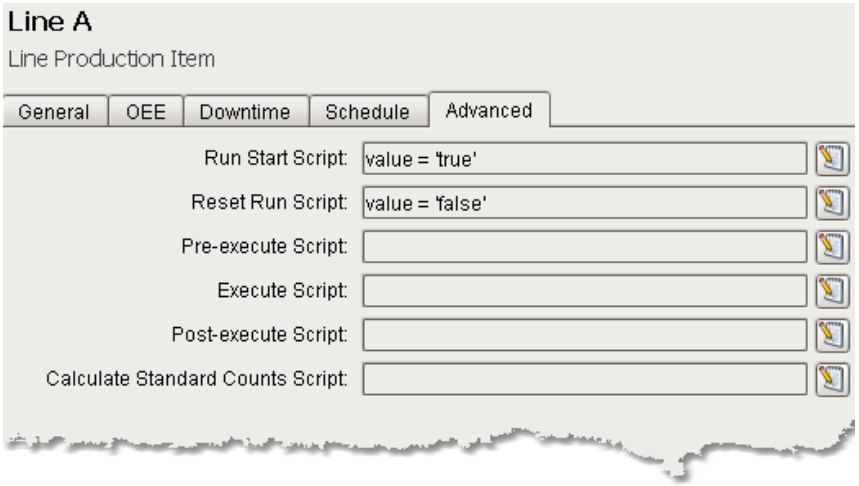
© Inductive Automation

information.

Additional scripting help and examples can be found in the Ignition Manual.

2.7.1 Line Events

Due to modules included in the OEE Downtime and Scheduling Module, the need for scripting is virtually nonexistent. However, if the user would like to expand on the existing scripting, or make adjustments to better fit his or her needs, this can still be done within Ignition. Scripting is also used with *Lines* and *Cells* under the **Advanced** tab. In order to edit the script under this tab, simply click the button and enter the desired script, then click **OK** to save.



Advanced Tab for a Line

These scripts are executed in relation to the line events:

Run Start Script - When a run is started. Triggered once before the pre-execute script.

Reset Run Script - When a run reset is triggered. This occurrs when a new schedule is selected or a new product code for a line is selected but, before the line starts actually running. Triggered once before the pre-execute script.

Pre-execute Script - Triggered before each time the line is to perform the process of updating the line counts, and calculating the OEE values.

Execute Script - Triggered every time after the line has executed the updates and calculations, but, before the values have been written to the database.

Post Execute Script - Triggered every time after the line has written the lates values to the database.

Calculate Standard Counts Script - The Calculate Standard Counts Script and allow the use a different method of OEE calculation. See example below.

Example:
This script is used under Run Start Script in the demo project and will cause the PLC production

simulator to run when the operator clicks *Start*.

```
value = 'true'
system.tag.writeToTag('[Default]Line 1/PLC/Run', value)
```

The Calculate Standard Counts Script and allow the use a different method of OEE calculation

Located in Line -> Advanced -> Calculate Standard Counts Script
Calculations are performed every minute. If you set a new value, it takes affect until the next time the calculations are performed.

OEECounts event

```
event.getPath() String
event.getName() String
event.getInfeedCount() Integer
event.getProductionCount() Integer
event.getPackageCount() Double
event.getStandardRate() Double
event.getStandardRatePeriod() String

event.getTargetCount() Integer
event.setTargetCount(Integer targetCount) void
event.getTargetVariance() Integer
event.setTargetVariance(Integer targetVariance) void
event.getStandardCount() Integer
event.setStandardCount(Integer standardCount) void
event.getIdealStandardCount() Integer
event.setIdealStandardCount(Integer idealStandardCount) void
event.getStandardVariance() Integer
event.setStandardVariance(Integer standardVariance) void
event.getWasteCount() Integer
event.setWasteCount(Integer wasteCount) void
```

2.7.2 Custom OEE Calculations

Created in Project -> Events Script (Gateway) -> Startup

If a custom script is added for any key then it is responsible for setting the value, internal calculations will not be run.
Each key type will allow access to the component parts of the calculation, see definition of each key for those parts.

Format:

system.oee.addCustomScript(key, script)
system.oee.removeCustomScript(key)

Key:

system.oee.AVAILABILITY
system.oee.PERFORMANCE
system.oee.QUALITY
system.oee.OEE

Event properties:
AVAILABILITY key

```
event.getElapsedTime() Double
event.getRunTime() Double
event.getPlannedDownTime() Double
event.getAvailability() Double
event.setAvailability(Double availability) void
```

PERFORMANCE event

```
event.getInfeedCount() Integer
event.getPackageCount() Double
event.getStandardRatePeriod() String
event.getStandardRate() Double
event.getProductionCount() Integer
event.getWasteCount() Integer
event.getRunTimeMin() Double
event.getElapsedTimeMin() Double
event.getStandardCount() Integer
event.getPerformance() Double
event.setPerformance(Double performance) void
```

QUALITY event

```
event.getInfeedCount() Integer
event.getWasteCount() Integer
event.getQuality() Double
event.void setQuality(Double quality) void
```

OEE event

```
event.getAvailability() Double
event.getPerformance() Double
event.getQuality() Double
event.getOEE() Double
event.setOEE(Double oee) void
```

Example:

```
script="event.setAvailability(0.85)"
system.oee.addCustomScript(system.oee.AVAILABILITY, script)

system.oee.removeCustomScript(system.oee.AVAILABILITY)
```

2.7.3 OEE Factor Cap

Created in Project -> Events Script (Gateway) -> Startup

Allows overriding the default cap of 100% for OEE factors of Availability, Performance and Quality.
If this script exists during a gateway save or restart then the OEE factor(s) will be set to the maxPercent.
If the script is removed and the gateway is restarted the OEE factor(s) will be set to 100%.

Format:
system.oee.setOEEFactorCap(key, maxPercent)

Key:
system.oee.AVAILABILITY
system.oee.PERFORMANCE
system.oee.QUALITY

Example:

```
system.oee.setOEEFactorCap(system.oee.AVAILABILITY, 250)
system.oee.setOEEFactorCap(system.oee.PERFORMANCE, 150)
system.oee.setOEEFactorCap(system.oee.QUALITY, 75)
```

2.7.4 Gateway Scripts

Methods

```
system.production.addProductCode(projectName, productCode, description)
```

Adds a new product code to the system.

parameters

`projectName`	A valid project name. `Data Type` String
`productCode`	The product code. For example: "Cola_12oz_Cans" `Data Type` String
`description`	A description of this product code. `Data Type` String

returns

`result`	Error message if the product code could not be added, blank if successful. `Data Type` String

```
system.production.utils.addRunComment(projectName, linePath, userName, note, isSticky
```

Adds a comment note to the current run for the selected line.

parameters

`projectName`	A valid project name. `Data Type` `String`
`linePath`	The line path of the production *line* that this component is associated with. This is the full path name of the *line* starting with the project name. For example: "OEEDemo\Your Enterprise\Your Site\Your Area\Line 1" `Data Type` `String`
`userName`	User name for this comment `Data Type` `String`
`note`	This comment `Data Type` `String`
`isSticky`	If set to 1 (True) the note will appear at the top of the list, if 0 (False) the note will appear in order it was entered. `Data Type` `Boolean`

returns

`none`

© Inductive Automation

```
system.production.adjustRunData(runUUID, cellName, factorName, factorValue)
system.production.adjustRunData(runUUID, cellName, factorName, factorValue,
adjustInfeed)
```

Recalculates production data for a line or a cell based on the factor name and the factor value. The run must be complete for adjustment to be accurate.

parameters

`runUUID`	The unique run identifier of the run to adjust. This value can be accessed via the Analysis Controller datapoint called "Run Identifier". `Data Type` `String`
`cellName`	Name of the cell to adjust. Leave blank to indicate the line `Data Type` `String`
`factorName`	The run factor to adjust. The possible values are "InfeedCount' "ProductionCount" or "WasteCount" `Data Type` `String`
`factorValue`	The value to set the factor to. NOTE: PRODUCTION DATA W BE MODIFIED AND CANNOT BE UNDONE. USE WITH EXTREME CAUTION. `Data Type` `Double`
`adjustInfeed`	If set to 1 (True) the InfeedCount will always be modified, if 0 (False) the InfeedCount will not be modified if ProductionCount or WasteCount are being adjusted. If this parameter is omitted the default is 1 (True). `Data Type` `Boolean`

returns

`none`

```
system.production.adjustRunDataByShift(runUUID, cellName, ShiftDate, factorName,
factorValue)
system.production.adjustRunDataByShift(runUUID, cellName, ShiftDate, factorName,
factorValue, adjustInfeed)
```

Recalculates production data for a line or a cell based on the shift, factor name and the factor value. The run must be complete for adjustment to be accurate.

parameters

`runUUID`	The unique run identifier of the run to adjust. This value can be accessed via the Analysis Controller datapoint called "Run Identifier". Data Type String
`cellName`	Name of the cell to adjust. Leave blank to indicate the line Data Type String
`shiftDate`	The shift date you want adjusted. This value can be accessed via the Analysis Controller datapoint called "Shift Date". Data Type Date
`factorName`	The run factor to adjust. The possible values are "InfeedCount", "ProductionCount" or "WasteCount" Data Type String
`factorValue`	The value to set the factor to. NOTE: PRODUCTION DATA WILL BE MODIFIED AND CANNOT BE UNDONE. USE WITH EXTREME CAUTION. Data Type Double
`adjustInfeed`	If set to 1 (True) the InfeedCount will always be modified, if 0 (False) the InfeedCount will not be modified if ProductionCount or WasteCount are being adjusted. If this parameter is omitted the default is 1 (True). Data Type Boolean

returns

`none`

```
system.production.cancelRun(projectName, linePath)
```

Cancel the current run for a line. This is only valid if the production run is currently in the changeover period.

parameters

`projectName`	The project name that contains the specified line path. Data Type String
`linePath`	The line path of the production *line* that this component is associated with. This is the full path name of the *line* starting with the project name. For example: "OEEDemo\Your Enterprise\Your Site\Your Area\Line 1" Data Type String

returns

`successful`	Returns true if the production run successfully has been canceled. Data Type Boolean

---

```
system.production.endRun(projectName, linePath)
```

End the current run for a line. This is only valid if the line is currently in a production run. After a production run has been ended, it can restarted using the resume script function.

parameters

`projectName`	The project name that contains the specified line path. Data Type String
`linePath`	The line path of the production *line* that this component is associated with. This is the full path name of the *line* starting with the project name. For example: "OEEDemo\Your Enterprise\Your Site\Your Area\Line 1" Data Type String

returns

`successful`	Returns true if the production run successfully ended. Data Type Boolean

---

```
system.production.getLineID(projectName, linePath)
```

Returns the internal line id of the given line path. Allows advanced usage of direct SQL queries in the database.

parameters

`projectName`	The project name that contains the specified line path. `Data Type` `String`
`linePath`	The line path of the production *line* that this component is associated with. This is the full path name of the *line* starting with the project name. For example: "OEEDemo\Your Enterprise\Your Site\Your Area\Line 1" `Data Type` `String`

returns

`lineID`	Returns the internal line id of the linepath or -1 if the line could not be found. `Data Type` `Integer`

```
system.production.isProductionModelRunning(projectName)
```

Returns true if the production model for this project is running.

parameters

`projectName`	The project name that contains the specified line path. `Data Type` `String`

returns

`running`	Returns true if the production model for this project is running. `Data Type` `Boolean`

```
system.production.reset(runUUID, cellName, shiftDate, factorName)
```

Sets to 0 the Infeed count and the factor count for the cell or line. The run must be complete for adjustment to be accurate.

parameters

`runUUID`	The unique run identifier of the run to reset. This value can be accessed via the Analysis Controller datapoint called "Run Identifier". `Data Type` `String`
`cellName`	Name of the cell to reset. Leave blank to indicate the line. `Data Type` `String`
`shiftDate`	The shift date you want reset. This value can be accessed via the Analysis Controller datapoint called "Shift Date". `Data Type` `Date`
`factorName`	The run factor to adjust. The possible values are "InfeedCount", "ProductionCount" or "WasteCount" `Data Type` `String`

returns

`none`

```
system.production.resumeRun(projectName, linePath)
```

Resume the current production run for a line. This is only valid if the production run has ended.

parameters

`projectName`	The project name that contains the specified line path. `Data Type` `String`
`linePath`	The line path of the production *line* that this component is associated with. This is the full path name of the *line* starting with the project name. For example: "OEEDemo\Your Enterprise\Your Site\Your Area\Line 1" `Data Type` `String`

returns

`successful`	Returns true if the production run successfully has been resumed. `Data Type` `Boolean`

```
system.production.setLineProductCode(projectName, linePath, productCode)
```

Set the current product code for a line. If the line is currently in a production run, it will have to be ended before setting a new product code. The product code must exist in the production code table and the line must be enabled to run it.

parameters

`projectName` — The project name that contains the specified line path.
Data Type: String

`linePath` — The line path of the production *line* that this component is associated with. This is the full path name of the *line* starting with the project name.
For example: "OEEDemo\Your Enterprise\Your Site\Your Area\Line 1"
Data Type: String

`productCode` — The new product code for the line to run next.
Data Type: String

returns

`successful` — Returns true if the project name, line path and product code are valid and the new product code has been set.
Data Type: Boolean

```
system.production.startLineProductCode(projectName, linePath, productCode)
```

Set the current product code for a line and immediately starts it running. If the line is currently in a production run, it will have to be ended before setting a new product code. The product code must exist in the production code table and the line must be enabled to run it.

parameters

`projectName` — The project name that contains the specified line path.
Data Type: String

`linePath` — The line path of the production *line* that this component is associated with. This is the full path name of the *line* starting with the project name.
For example: "OEEDemo\Your Enterprise\Your Site\Your Area\Line 1"
Data Type: String

`productCode` — The new product code for the line to run next.
Data Type: String

returns

`successful` — Returns true if the project name, line path and product code are valid and the new product code has been set.
Data Type: Boolean

```
system.production.startRun(projectName, linePath)
```

Start a new production run for the current product code. This is only valid if the line is not currently in a production run.

parameters

`projectName`	The project name that contains the specified line path. `Data Type` `String`
`linePath`	The line path of the production *line* that this component is associated with. This is the full path name of the *line* starting with the project name. For example: "OEEDemo\Your Enterprise\Your Site\Your Area\Line 1" `Data Type` `String`

returns

`successful`	Returns true if the production run successfully started. `Data Type` `Boolean`

```
system.production.updateProductCodeLineStatus(projectName, productCode, linePath,
enable)
```

Updates the line enabled status for this product code.

parameters

`projectName`	The project name that contains the specified line path. `Data Type` `String`
`linePath`	The line path of the production *line* that this component is associated with. This is the full path name of the *line* starting with the project name. For example: "OEEDemo\Your Enterprise\Your Site\Your Area\Line 1" `Data Type` `String`
`productCode`	The new product code for the line to run next. `Data Type` `String`
`enable`	Set to 0 to disable, 1 to enable. `Data Type` `Integer`

© Inductive Automation

```
system.schedule.selectRun(projectName, linePath, scheduleID)
```

Select the schedule entry to run on the specified line. The scheduleID can be obtained from the Schedule database table. If the line is currently in a production run, it will have to be ended before setting a new product code. The schedule entry must be valid with a work order and product code appropriate for the line.

parameters

`projectName` — The project name that contains the specified line path.
Data Type String

`linePath` — The line path of the production *line* that this component is associated with. This is the full path name of the *line* starting with the project name.
For example: "OEEDemo\Your Enterprise\Your Site\Your Area\Line 1"
Data Type String

`scheduleID` — The value from the ID column of the schedule database table
Data Type Integer

returns

`successful` — Returns true if the new schedule entry for the line has successfully been selected.
Data Type Boolean

```
system.schedule.selectNextRun(projectName, linePath)
```

Select the next schedule entry to run on the specified line. The next schedule entry is the row in the database Schedule table in chronological order by the StartDateTime column. The schedule entry must be valid with a work order and product code appropriate for the line.

parameters

`projectName` — The project name that contains the specified line path.
Data Type String

`linePath` — The line path of the production *line* that this component is associated with. This is the full path name of the *line* starting with the project name.
For example: "OEEDemo\Your Enterprise\Your Site\Your Area\Line 1"
Data Type String

returns

`successful` — Returns true if the new schedule entry for the line has successfully been selected.
Data Type Boolean

2.7.5 Client/Designer Scripts

Methods

```
system.production.utils.addProductCode(productCode, description)
```

Adds a new product code to the system.

parameters

`productCode`	The product code. For example: "Cola_12oz_Cans" `Data Type`	`String`
`description`	A description of this product code. `Data Type`	`String`

returns

`result`	Error message if the product code could not be added, blank if successful. `Data Type`	`String`

```
system.production.utils.addRunComment(linePath, userName, note, isSticky
```

Adds a comment note to the current run for the selected line.

parameters

`linePath`	The line path of the production *line* that this component is associated with. This is the full path name of the *line* starting with the project name. For example: "OEEDemo\Your Enterprise\Your Site\Your Area\Line 1" `Data Type`	`String`
`userName`	User name for this comment `Data Type`	`String`
`note`	This comment `Data Type`	`String`
`isSticky`	If set to 1 (True) the note will appear at the top of the list, if 0 (False) the note will appear in order it was entered. `Data Type`	`Boolean`

returns

`none`

```
system.production.utils.adjustRunData(runUUID, cellName, factorName, factorValue)
system.production.utils.adjustRunData(runUUID, cellName, factorName, factorValue,
adjustInfeed)
```

Recalculates production data for a line or a cell based on the factor name and the factor value. The run must be complete for adjustment to be accurate.

parameters

`runUUID`	The unique run identifier of the run to adjust. This value can be accessed via the Analysis Controller datapoint called "Run Identifier". `Data Type` String
`cellName`	Name of the cell to adjust. Leave blank to indicate the line `Data Type` String
`factorName`	The run factor to adjust. The possible values are "InfeedCount", "ProductionCount" or "WasteCount" `Data Type` String
`factorValue`	The value to set the factor to. NOTE: PRODUCTION DATA WILL BE MODIFIED AND CANNOT BE UNDONE. USE WITH EXTREME CAUTION. `Data Type` Double
`adjustInfeed`	If set to 1 (True) the InfeedCount will always be modified, if 0 (False) the InfeedCount will not be modified if ProductionCount or WasteCount are being adjusted. `Data Type` Boolean

returns

`none`

```
system.production.utils.adjustRunDataByShift(runUUID, cellName, ShiftDate,
factorName, factorValue)
system.production.utils.adjustRunDataByShift(runUUID, cellName, ShiftDate,
factorName, factorValue, adjustInfeed)
```

Recalculates production data for a line or a cell based on the shift, factor name and the factor value. The run must be complete for adjustment to be accurate.

parameters

`runUUID`	The unique run identifier of the run to adjust. This value can be accessed via the Analysis Controller datapoint called "Run Identifier". `Data Type` `String`
`cellName`	Name of the cell to adjust. Leave blank to indicate the line `Data Type` `String`
`shiftDate`	The shift date you want adjusted. This value can be accessed via the Analysis Controller datapoint called "Shift Date". `Data Type` `Date`
`factorName`	The run factor to adjust. The possible values are "InfeedCount", "ProductionCount" or "WasteCount" `Data Type` `String`
`factorValue`	The value to set the factor to. NOTE: PRODUCTION DATA WILL BE MODIFIED AND CANNOT BE UNDONE. USE WITH EXTREME CAUTION. `Data Type` `Double`
`adjustInfeed`	If set to 1 (True) the InfeedCount will always be modified, if 0 (False) the InfeedCount will not be modified if ProductionCount or WasteCount are being adjusted. If this parameter is omitted the default is 1 (True). `Data Type` `Boolean`

returns

`none`

`system.production.utils.cancelRun(linePath)`

Cancel the current run for a line. This is only valid if the production run is currently in the changeover period.

parameters

`linePath`	The line path of the production *line* that this component is associated with. This is the full path name of the *line* starting with the project name. For example: "OEEDemo\Your Enterprise\Your Site\Your Area\Line 1" Data Type String

returns

`successful`	Returns true if the production run successfully has been canceled. Data Type Boolean

---

`system.production.utils.endRun(linePath)`

End the current run for a line. This is only valid if the line is currently in a production run. After a production run has been ended, it can restarted using the resume scriptioning function.

parameters

`linePath`	The line path of the production *line* that this component is associated with. This is the full path name of the *line* starting with the project name. For example: "OEEDemo\Your Enterprise\Your Site\Your Area\Line 1" Data Type String

returns

`successful`	Returns true if the production run successfully ended. Data Type Boolean

---

`system.production.utils.getLineID(linePath)`

Returns the internal line id of the given line path. Allows advanced usage of direct SQL queries in the database.

parameters

	`linePath`	The line path of the production *line* that this component is associated with. This is the full path name of the *line* starting with the project name. For example: "OEEDemo\Your Enterprise\Your Site\Your Area\Line 1" `Data Type` `String`

returns

	`lineID`	Returns the internal line id of the linepath or -1 if the line could not be found. `Data Type` `Integer`

`system.production.utils.isProductionModelRunning()`

Returns true if the production model for this project is running.

parameters

(none)

returns

	`running`	Returns true if the production model for this project is running. `Data Type` `Boolean`

```
system.production.utils.reset(runUUID, cellName, shiftDate, factorName)
```

Sets to 0 the Infeed count and the factor count for the cell or line. The run must be complete for adjustment to be accurate.

parameters

runUUID	The unique run identifier of the run to reset. This value can be accessed via the Analysis Controller datapoint called "Run Identifier". Data Type String
cellName	Name of the cell to reset. Leave blank to indicate the line. Data Type String
shiftDate	The shift date you want reset. This value can be accessed via the Analysis Controller datapoint called "Shift Date". Data Type Date
factorName	The run factor to adjust. The possible values are "InfeedCount", "ProductionCount" or "WasteCount" Data Type String

returns

none

```
system.production.utils.resumeRun(linePath)
```

Resume the current production run for a line. This is only valid if the production run has been ended.

parameters

linePath	The line path of the production *line* that this component is associated with. This is the full path name of the *line* starting with the project name. For example: "OEEDemo\Your Enterprise\Your Site\Your Area\Line 1" Data Type String

returns

successful	Returns true if the production run successfully has been resumed. Data Type Boolean

---

`system.production.utils.setLineProductCode(linePath, productCode)`

Set the current product code for a line. If the line is currently in a production run, it will have to be ended before setting a new product code. The product code must exist in the production code table and the line must be enabled to run it.

parameters

`linePath`	The line path of the production *line* that this component is associated with. This is the full path name of the *line* starting with the project name. For example: "OEEDemo\Your Enterprise\Your Site\Your Area\Line 1" `Data Type` `String`
`productCode`	The new product code for the line to run next. `Data Type` `String`

returns

`successful`	Returns true if the project name, line path and product code are valid and the new product code has been set. `Data Type` `Boolean`

`system.production.utils.startLineProductCode(linePath, productCode)`

Set the current product code for a line and immediately starts it running. If the line is currently in a production run, it will have to be ended before setting a new product code. The product code must exist in the production code table and the line must be enabled to run it.

parameters

`linePath`	The line path of the production *line* that this component is associated with. This is the full path name of the *line* starting with the project name. For example: "OEEDemo\Your Enterprise\Your Site\Your Area\Line 1" `Data Type` `String`
`productCode`	The new product code for the line to run next. `Data Type` `String`

returns

`successful`	Returns true if the project name, line path and product code are valid and the new product code has been set. `Data Type` `Boolean`

`system.production.utils.startRun(linePath)`

Start a new production run for the current product code. This is only valid if the line is not currently in a production run.

parameters

`linePath`	The line path of the production *line* that this component is associated with. This is the full path name of the *line* starting with the project name. For example: "OEEDemo\Your Enterprise\Your Site\Your Area\Line 1" Data Type String

returns

`successful`	Returns true if the production run successfully started. Data Type Boolean

---

`system.production.utils.updateProductCodeLineStatus(productCode, linePath, enable)`

Updates the line enabled status for this product code.

parameters

`linePath`	The line path of the production *line* that this component is associated with. This is the full path name of the *line* starting with the project name. For example: "OEEDemo\Your Enterprise\Your Site\Your Area\Line 1" Data Type String
`productCode`	The new product code for the line to run next. Data Type String
`enable`	Set to 0 to disable, 1 to enable. Data Type Integer

`system.schedule.selectRun(linePath, scheduleID)`

Select the schedule entry to run on the specified line. The scheduleID can be obtained from the Schedule database table. If the line is currently in a production run, it will have to be ended before setting a new product code. The schedule entry must be valid with a work order and product code appropriate for the line.

parameters

`linePath`	The line path of the production *line* that this component is associated with. This is the full path name of the *line* starting with the project name. For example: "OEEDemo\Your Enterprise\Your Site\Your Area\Line 1" `Data Type` `String`
`scheduleID`	The value from the ID column of the schedule database table `Data Type` `Integer`

returns

`successful`	Returns true if the new schedule entry for the line has successfully been selected. `Data Type` `Boolean`

`system.schedule.selectNextRun(linePath)`

Select the next schedule entry to run on the specified line. The next schedule entry is the row in the database Schedule table in chronological order by the StartDateTime column. The schedule entry must be valid with a work order and product code appropriate for the line.

parameters

`linePath`	The line path of the production *line* that this component is associated with. This is the full path name of the *line* starting with the project name. For example: "OEEDemo\Your Enterprise\Your Site\Your Area\Line 1" `Data Type` `String`

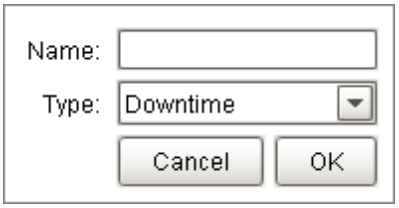
returns

`successful`	Returns true if the new schedule entry for the line has successfully been selected. `Data Type` `Boolean`

2.8 Analysis Providers

Analysis providers determine which information will be viewed on a graph or pie chart. Based on which Analysis Provider is selected, some filter, compare by, and data point options may or may not be visible. For example, the filter **Recordable Downtime** can be selected if the analysis provider is Downtime, but not if the analysis provider is Comment.

© Inductive Automation



Analysis Providers

2.8.1 Comment

Description

The Comment Analysis Provider is used to query production run comments entered by users.

Provider Name

Comment

Filters

These are the filters that are available in the OEE Downtime and Scheduling Module. However, in addition to these filters, additional factors may be available if they are string data type. All additional factors start with "Factor:". For example, "Factor:Operator". A filter will allow the user to see all of the data points in the analysis provider as it pertains to a specific *area*, shift, etc. For more information on filters, see the Filter By paragraph in the Analysis Screen section.

Area

Enterprise

Line

Package Count

Product Code

Production Units

Run

Shift

Site

Compare By

These are the comparisons that are available in the OEE Downtime and Scheduling Module. However, in addition to these comparisons, additional factors may be available if they are string data type. All additional factors start with "Factor:". For example, "Factor:Operator". A comparison allows one data point to be compared between all *areas*, days, etc. For more information on comparisons, see the Compare By paragraph in the Analysis Screen section.

Area

Day

Enterprise

Line

Month

Package Count

© Inductive Automation

Product Code

Production Units

Run

Shift

Site

Week

Data Points

These are the data points that are available in the OEE Downtime and Scheduling Module. However, in addition to these comparisons, additional factors may be available if they are string data type. All additional factors start with "Factor:". For example, "Factor:Operator". Data points are the different values that will be presented or compared on a graph or chart. For more information on data points, see the Data Point paragraph in the Analysis Screen section.

Area

Comment

Date

Entered By

Enterprise

Line

Package Count

Product Code

Product Code Description

Production Units

Run

Shift

Site

2.8.2 Downtime

Description

The Downtime Analysis Provider is used to analyze downtime data.

Provider Name

Downtime

Filters

These are the filters that are available in the OEE Downtime and Scheduling Module. However, in addition to these filters, additional factors may be available if they are string data type. All additional factors start with "Factor:". For example, "Factor:Operator". A filter will allow the user to see all of the data points in the analysis provider as it pertains to a specific *area*, shift, etc. For more information on filters, see the Filter By paragraph in the Analysis Screen section.

© Inductive Automation

Area

Automatic Reason

Cell Name

Enterprise

Line

Operator Reason

Package Count

Planned Downtime

Product Code

Production Units

Recordable Downtime

Run

Shift

Site

Compare By

These are the comparisons that are available in the OEE Downtime and Scheduling Module. However, in addition to these comparisons, additional factors may be available if they are string data type. All additional factors start with "Factor:". For example, "Factor:Operator". A comparison allows one data point to be compared between all *areas*, days, etc. For more information on comparisons, see the Compare By paragraph in the Analysis Screen section.

Area

Automatic Reason

Cell Name

Enterprise

Line

Operator Reason

Package Count

Product Code

Production Units

Run

Shift

Site

Data Points

These are the data points that are available in the OEE Downtime and Scheduling Module. However, in addition to these comparisons, additional factors may be available if they are string data type. All additional factors start with "Factor:". For example, "Factor:Operator". Data points are the different values that will be presented or compared on a graph or chart. For more information on data points, see the Data

© Inductive Automation

Point paragraph in the Analysis Screen section.

Area

Automatic Reason

Cell Name

Enterprise

Line

Occurrences

Operator Reason

Package Count

Product Code

Production Units

Run

Shift

Site

2.8.3 OEE

Description

The Run Analysis Provider is used to analyze OEE and production data.

Provider Name

Run

© Inductive Automation

Filters

These are the filters that are available in the OEE Downtime and Scheduling Module. However, in addition to these filters, additional factors may be available if they are string data type. All additional factors start with "Factor:". For example, "Factor:Operator". A filter will allow the user to see all of the data points in the analysis provider as it pertains to a specific *area*, shift, etc. For more information on filters, see the Filter By paragraph in the Analysis Screen section.

Area

Cell Name

Enterprise

Hour Of Run

Line

Package Count

Product Code

Production Units

Run

Shift

Site

Compare By

These are the comparisons that are available in the OEE Downtime and Scheduling Module. However, in addition to these comparisons, additional factors may be available if they are string data type. All additional factors start with "Factor:". For example, "Factor:Operator". A comparison allows one data point to be compared between all *areas*, days, etc. For more information on comparisons, see the Compare By paragraph in the Analysis Screen section.

Area

Cell Name

Day

Enterprise

Hour Of Run

Line

Month

Package Count

Product Code

Production Units

Run

Shift

Site

Week

© Inductive Automation

Data Points

These are the data points that are available in the OEE Downtime and Scheduling Module. However, in addition to these comparisons, additional factors may be available if they are string data type. All additional factors start with "Factor:". For example, "Factor:Operator". Data points are the different values that will be presented or compared on a graph or chart. For more information on data points, see the Data Point paragraph in the Analysis Screen section.

Area

Cell Down Time

Cell Infeed Count

Cell Name

Cell Package Count

Cell Production Count

Cell Production Units

Cell Run Time

Cell Standard Count

Cell Target Count

Cell Waste Count

Date

Enterprise

Hour Of Run

Line

Line Infeed Count

Line Production Count

Line Standard Count

Line Standard Rate

Line Standard Rate Period

Line Target Count

Line Waste Count

OEE

OEE Availability

OEE Performance

OEE Quality

Package Count

Product Code

Production Units

Run

Run Down Time

© Inductive Automation

Run Elapsed Time

Run Planned Down Time

Run Time

Shift

Site

2.8.4 Schedule

Description

The Schedule Analysis Provider is used to analyze scheduled versus actual production run times.

Provider Name

Schedule

Filters

These are the filters that are available in the OEE Downtime and Scheduling Module. However, in addition to these filters, additional factors may be available if they are string data type. All additional factors start with "Factor:". For example, "Factor:Operator". A filter will allow the user to see all of the data points in the analysis provider as it pertains to a specific area, shift, etc. For more information on filters, see the Filter By paragraph in the Analysis Screen section.

Area

Enterprise

Line

Package Count

Product Code

Production Units

Run

Shift

Site

Compare By

These are the comparisons that are available in the OEE Downtime and Scheduling Module. However, in addition to these comparisons, additional factors may be available if they are string data type. All additional factors start with "Factor:". For example, "Factor:Operator". A comparison allows one data point to be compared between all areas, days, etc. For more information on comparisons, see the Compare By paragraph in the Analysis Screen section.

Area

Enterprise

Line

Package Count

Product Code

© Inductive Automation

Production Units

Site

Data Points

These are the data points that are available in the OEE Downtime and Scheduling Module. However, in addition to these comparisons, additional factors may be available if they are string data type. All additional factors start with "Factor:". For example, "Factor:Operator". Data points are the different values that will be presented or compared on a graph or chart. For more information on data points, see the Data Point paragraph in the Analysis Screen section.

Actual Finish Time

Actual Run Start Time

Actual Start Time

Area

Enterprise

Line

Package Count

Product Code

Product Code Description

Production Units

Run

Scheduled Finish Time

Scheduled Quantity

Scheduled Run Start Time

Scheduled Start Time

Site

2.8.5 TEEP

Description

The TEEP Analysis Provider is used to analyze utilization data.

Provider Name

TEEP

© Inductive Automation

Filters

These are the filters that are available in the OEE Downtime and Scheduling Module. However, in addition to these filters, additional factors may be available if they are string data type. All additional factors start with "Factor:". For example, "Factor:Operator". A filter will allow the user to see all of the data points in the analysis provider as it pertains to a specific *area*, shift, etc. For more information on filters, see the Filter By paragraph in the Analysis Screen section.

Area

Enterprise

Line

Package Count

Product Code

Production Units

Run

Site

Compare By

These are the comparisons that are available in the OEE Downtime and Scheduling Module. However, in addition to these comparisons, additional factors may be available if they are string data type. All additional factors start with "Factor:". For example, "Factor:Operator". A comparison allows one data point to be compared between all *areas*, days, etc. For more information on comparisons, see the Compare By paragraph in the Analysis Screen section.

Area

Enterprise

Line

Package Count

Product Code

Production Units

Run

Site

Data Points

These are the data points that are available in the OEE Downtime and Scheduling Module. However, in addition to these comparisons, additional factors may be available if they are string data type. All additional factors start with "Factor:". For example, "Factor:Operator". Data points are the different values that will be presented or compared on a graph or chart. For more information on data points, see the Data Point paragraph in the Analysis Screen section.

Area

Enterprise

Line

Loading (Actual)

© Inductive Automation

Loading (Scheduled)

OEE

OEE Availability

OEE Performance

OEE Quality

Package Count

Product Code

Production Units

Run

Shift

Site

TEEP (Actual)

TEEP (Scheduled)

2.9 Miscellaneous

This section contains additional information to be used for reference.

2.9.1 Additional Factors

The OEE Downtime and Scheduling Module collects and logs a number of downtime and production data values. However, what if other values outside of downtime and production values are of interest? Additional factors are the solution.

Additional Factors are user defined data points that are logged along with the production and downtime information. Once they are logged, they can be shown in charts, tables and reports. Additionally, other analyses can be done by filtering and/or setting up comparisons by their values.

Any value that can be read from an Ignition SQLTag can be added as a additional factor. This includes values derived from scripts, or from barcode readers, databases, calculations, PLCs, etc.

Example: An additional factory named cardboard manufacturer can be added. The operator can select the manufacturer that provided the cardboard or it can be obtained from some other source. Now, OEE and downtime results can be shown for each cardboard manufacturer. This can identify quality problems with raw material that directly affect efficiencies.

In the OEEDemo, the operator is setup as an additional factor. The operator's name will be logged along with the production and downtime data. By doing so, OEE and downtime information can be filtered and grouped by the operator name. But this could just as well be the production crew, supervisor, maintenance crew or any other user defined value that can be monitored or entered into the system.

2.9.2 Production Rate Calculation

Rate per Minute

The production rate per minute is calculated from the change between the current production count and the production count from the prior minute.

© Inductive Automation

Rate per Hour

The production rate per hour is calculated by recording the production count every minute. Then the hourly rate is calculated from the change between the current production count and the production count from an hour ago. When a production *line* first starts up and there are no production counts from one hour ago, a project calculation is used.

© Inductive Automation

SPC Quality

Part III

inductive automation
800.266.7798
www.inductiveautomation.com

3 SPC Quality

This section of the manual is intended for documenting the SPC Module and does not expand on SPC (Statistical Process Control) itself. There are many books that go into great detail of both quality and SPC that should be referenced for information and procedures on determining how to improve quality.

Even though quality and SPC are sometimes used interchangeably, they are different. Quality is very broad and includes much more than just SPC, while SPC is used as a tool in the quality process. The Ignition SPC module focuses only on SPC.

A quick example may help with pointing out the difference between quality and SPC. If product in the warehouse is going bad over time, then a process has to start to narrow in on the cause. It will involve brainstorming and fishbone diagrams to determine the possible causes which may be the source of the problem. In the case of an off-color product, it could be rust building up in pipes, chemical formulation changes, or different raw materials being used. This part of the example refers to quality. Unlike SPC, quality requires more than installing software, collecting samples and analyzing the results.

Once the most likely causes of the off-color product have been determined, SPC can be used to monitor the attributes and narrow down and isolate the cause. It may be determined that when the pH of a sub-ingredient falls out of a certain range, the stability of the product color is degraded. With this knowledge, SPC can be used to monitor the pH so that if it falls out side of range, it can be corrected quickly. This prevents a bigger problem that may appear after the product sits in the warehouse for a period of time.

3.1 Introduction

The Ignition SPC module exceeds the capabilities of normal SPC software. It performs many tasks beyond control charts and manual data entry. Below is a list of some of the features that the SPC modules is capable of:

- Manual sample collections
- Automatic sample collections
- Scheduling samples based on realtime production conditions
- Alerting of samples coming due, due or overdue
- Automatic evaluation of control limits and out of control signals without human intervention
- Alerting of out of control conditions
- Customizable screens
- An much more

The SPC module is very powerful, but some implementations need more functionality or different functionality than what is originally included. The SPC module sits on top of the Ignition platform, which allows for configuring it to accommodate the desired functionality.

3.1.1 SPC Charts

Here is a brief overview of the control charts in the Quality modules. The control charts can be separated into three groups: value charts, attribute charts, and analysis charts. On all charts, it is possible to add assignable causes and notes to explain a data point. A sample note can be entered on the Lab or Test Stations page when the sample is first entered. This can be done by selecting a sample, then clicking Add Note. An attribute note is added directly from an SPC chart by right-clicking on a data point and selecting Set Note from the drop-down list. In addition to attribute notes, an assignable cause can also be added in this way. Assignable causes can also be saved for future use.

© Inductive Automation

Out-of-Control Signals and Control Limits can also be added to the graphs.

Value Charts

X Bar Range Chart: The X Bar Range Chart is used when there are multiple measurements taken in one sample. For example, if the pH is taken for five different pieces of product, the five different measurements will show up in the X Bar Range table. If all of these values are added together and then divided by the number of measurements taken, it will equal the average value, or x bar. This is what is graphed on the X Bar chart. When the lowest value is subtracted from the highest value, this equals the range, which is graphed on the Range chart. The range shows the overall consistency of the attribute being measured. The larger the range is, the less consistent the measurements are. If a point is not consistent with the rest of the data and is affecting other calculated values, this data point can be deleted. This will allow other calculated values, such as the x-double-bar and control limits, to reflect the data more accurately. X-double-bar is the average of all the averages, or the average of all the data points shown on the graph. Control limits are calculated to show where most data points on the graph will fall, provided the process is not out-of-control.

The X Bar Range chart should be used when data is generated frequently and is variable. This chart is useful for detecting small changes in the process and when multiple measurements are taken to represent a larger group of product.

Individuals Chart: The Individuals Chart is similar to the X Bar Range Chart. However, only one measurement is taken per sample instead of multiple. This means that the X Bar will always be the same value as the measurement, and a moving range will be calculated instead of the basic range. This means that instead of subtracting the lowest value from the highest value in one sample, moving range will calculate the difference between one sample and the next, showing the change from sample to sample. If a single measurement is used on the X Bar Range Chart, the range will always be zero, which fails to show the consistency between measurements.

Individuals charts are useful in situations when testing of a product results in the destruction of the product or if the testing is time consuming. It can also be used when a sample will yield the same result for a long period of time no matter how many measurements are made, such as batch operations. When using the Individuals Chart, the variable data should fall into a normal distribution, meaning the data points are equally likely to fall on either side of the average. This would appear as a bell curve on a histogram.

Median Chart: The Median Chart is also known as the MA-MR Chart or Moving Average-Moving Range Chart. Because data is generated slowly, the data on this chart is displayed differently. The first sample will contain three new data points. The second sample will contain the two most recent data points from sample one, in addition to one new data point. Sample three will contain the two most recent from sample two, as well as one new data point, and so on. Even though there are three samples with three data points each, there is only a total of five data points. Ont his chart, the median and the moving range are graphed. The median is the middle value based on the measurements in the sample (this is not the same as the average), while the range is the highest value minus the lowest value for each sample.

Like an individual chart, this chart should be used when the data is variable. In addition, data may also be costly or time-consuming to gather, or remain constant for a long periods of time. This chart should also be used when the data will not be normally distributed or when detecting small process changes.

X Bar Standard Deviation Chart: This chart is very similar to the X Bar Range Chart. The major difference between the two is that the X Bar and S chart uses standard deviation to find the amount of variation within a sample instead of the range.

© Inductive Automation

Data must be in variable form to use this chart. It should also be used when data is plentiful enough that samples can have ten measurements or more, or when there is a need to rapidly detect small changes.

Attribute Charts

P Chart: P charts are used to track the proportion of nonconforming items in a sample. The number of nonconformities per item is irrelevant for this type of chart, which only tracks the total number of items; however, it is possible to have the types of nonconformities displayed on the same chart.

P charts are used only when looking at the number of nonconforming items and when the sample size is not consistent.

NP Chart: Unlike the P Chart, the NP chart requires that all the sample sizes are the same. The number of nonconforming items is graphed instead of the proportion because the samples can be directly compared. The types of nonconformities can also be displayed on the same chart.

This chart should be used when counting nonconforming items when the sample size does not change.

C Chart: Also know as a count chart, this type counts the total number of nonconformities (not nonconforming items) on all the items in a sample. Often, the types of nonconformities and their individual counts are noted as well.

This chart is best used when counting nonconformities when the sample size will not vary. It is also important that each sample has equal opportunity for nonconformities.

U Chart: A U Chart also graphs the number of nonconformities, but does so through a proportion. In this chart, the types and counts of nonconformities are tracked as well.

This chart should be used when counting nonconformities when the sample size will vary. Also, if some samples have a greater opportunity for nonconformities than others, this chart should be used over the C Chart.

Analysis Charts

Histogram: A histogram shows the distribution of the data provided from the samples. A typical histogram has a "normal distribution," meaning that most data points will fall in the middle of the graph and fewer will fall towards the outside, forming a bell curve. A distribution that is normal is just the most common pattern. There are other types of curves, such as skewed distribution or double-peaked distribution, which may be typical for certain processes. If a bell-shaped curve is formed on the histogram, then any variations in the data are most likely due to an assignable cause. Assignable causes influence variations, which can occur in materials, environment, machines, peoples, etc. Ultimately, the histogram shows the consistency of a process.

Histograms should be used when data is numerical and the shape of the distribution is to be observed. Observing the shape of the graph can help to determine whether or not the data is distributed normally, if a change has occurred in the process over time, or if two or more processes are different. This graph can also help to communicate with others about the data distribution or determine if a process will be able to meet the requirements of a customer.

Pareto: The pareto chart is a bar chart that is used to show which factors are the biggest problems. The bars are arranged so that the most significant factor, that is, the factor that occurs the most frequently or cost the most (whether that be in time or money), is on the left, while the shortest bar, or least significant,

© Inductive Automation

is on the right.

Because of the organization of the pareto chart, it is best used when looking at how often problems or causes occur and which of those are the most significant, or when looking at a specific component of a larger problem. Like the histogram chart, the pareto is also useful for the communication of data.

3.1.2 Scheduling Samples

If you worry about samples being taken at the correct time and not being faked after the fact, you are not alone. It is not a matter of whether or not the person responsible for taking samples has been distracted and missed taking samples, it is a matter of when. The Ignition SPC module has powerful features that will schedule samples based on current realtime production conditions.

For example, if a lab staff is required to take samples every hour a production line is running, what happens when there is a break down or the production start is delayed because the lack of raw materials? How does the lab technician know when production started and if it has been a hour? In a variety of ways, the Ignition module can let the lab technician know that production has started and a sample is coming due, is due or is overdue. This can be expanded to instantly inform all parties that should know of various sample due states.

This can be utilized for more than taking live process samples. It can also be used for other checks that have to be done around the production facility such as weekly inspections of values or rodent traps.

3.1.3 Evaluating Signals

Typically, SPC software requires that someone opens a screen and visually checks for out of control conditions. Just like the scheduling of samples, someone may be distracted by other pressing production issues and fail to complete the task. The Ignition SPC module has powerful features that will automatically evaluate out of control signals every time new sample data is recorded. This can be expanded to instantly inform all parties that should know of various out of control conditions.

3.2 Getting Started

This getting started guide will step you though SPC Quality module installation, demo installation, the demo user interface and configuration features.

3.2.1 Installation

To install the SPC module into an existing Ignition system, follow the instructions in the Existing Ignition System. If you are installing Ignition at the same time, use the instructions in the New Ignition System.

To install the Quality Demo project, follow the steps in the Demo Installation section.

3.2.1.1 Existing Ignition System

3.2.1.1.1 Installing Modules

To install the SPC module on to an existing Ignition server, follow the steps below:

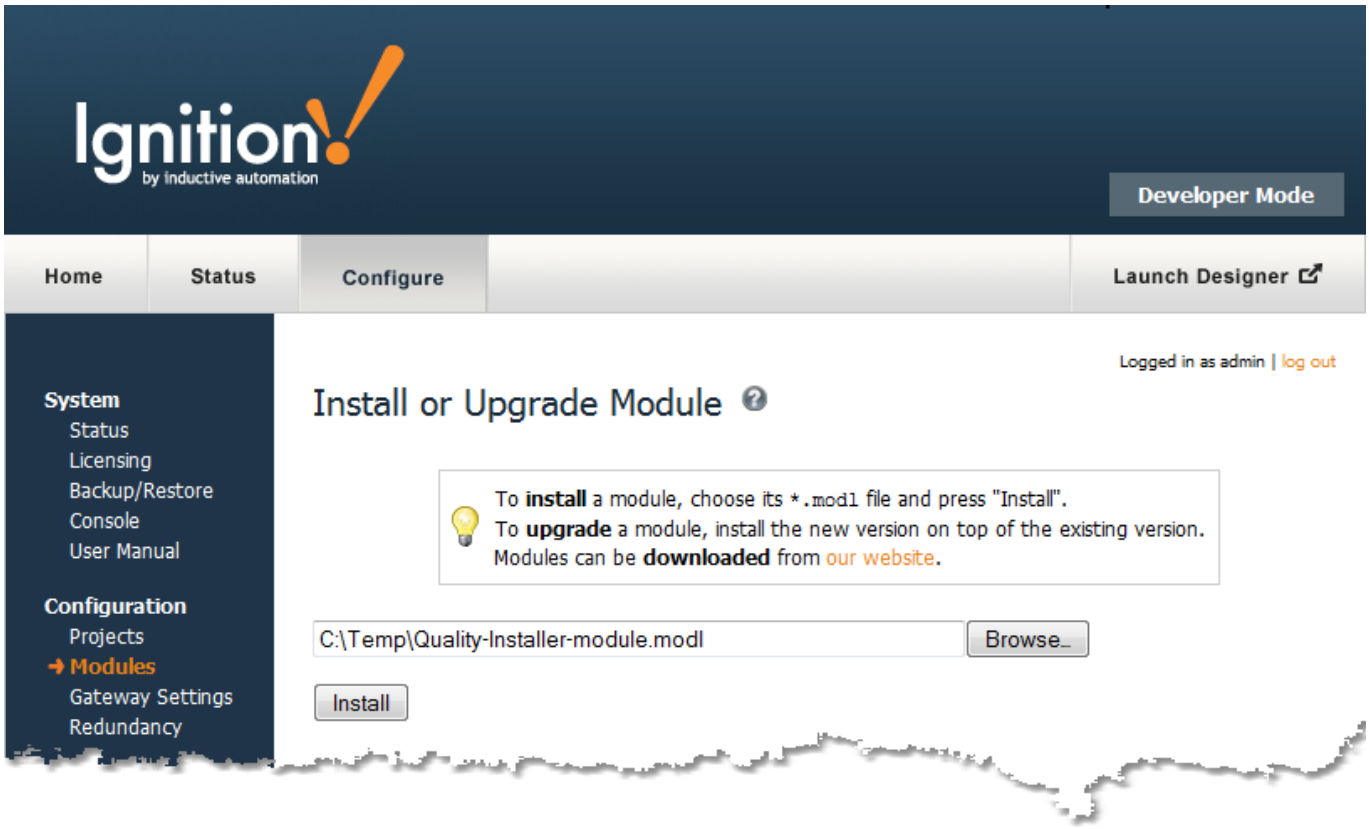
Before installing the SPC module, it is recommended to first setup the database connection that will be used to store SPC data.

1. **Download the Quality-Installer-module.modl module**
   from the Inductive Automation download website. It will be under the MES modules heading.

2. **Install the Quality-Installer-module.modl module**

© Inductive Automation

Navigate to the Modules page of the Ignition gateway. At the bottom of the list of already installed modules, click the Install or Upgrade a Module... link. Next, browse to the Quality-Installer-module.modl file and click the install button as shown below.



Install Ignition Module

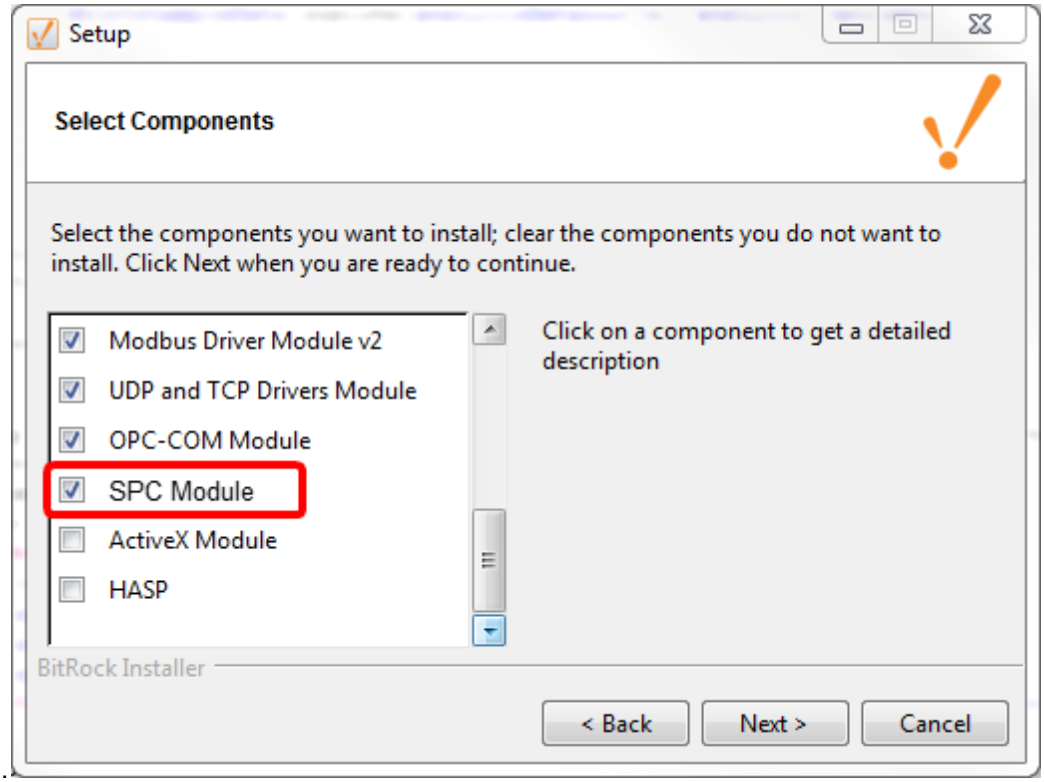
The SPC Installer module will install all required modules. These are the Production and SPC modules. It is important to keep in mind not to install or update these modules individually. Instead, it should be done by updating the SPC Installer module.

3.2.1.2 New Ignition System

3.2.1.2.1 Selecting Install Options

To install the SPC module at the same time as Ignition add the following steps to the normal Ignition installation:

1. **Select "Custom Configuration" on the setup step during the Ignition installation.**
   The following screen will appear. Scroll down to SPC Module and select it. This will cause the modules required for SPC functionality to be installed at the same time as Ignition.

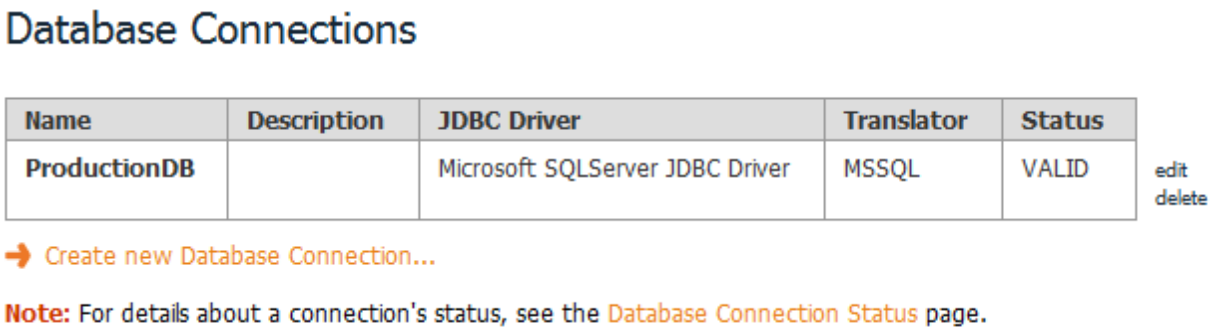

Setup
Select Components
Select the components you want to install; clear the components you do not want to install. Click Next when you are ready to continue.
Modbus Driver Module v2
UDP and TCP Drivers Module
OPC-COM Module
SPC Module
ActiveX Module
HASP
Click on a component to get a detailed description
BitRock Installer
< Back
Next >
Cancel

Ignition Installer

3.2.1.3 Configure Database

SPC data is stored in databases external to Ignition. These database(s) are setup in the gateway configuration section by selecting the **Databases> Connections** section from the left-hand configuration menu. See the Ignition documentation for more information on setting up a database connection. Below shows a typical database connection that is required for the SPC module.

© Inductive Automation



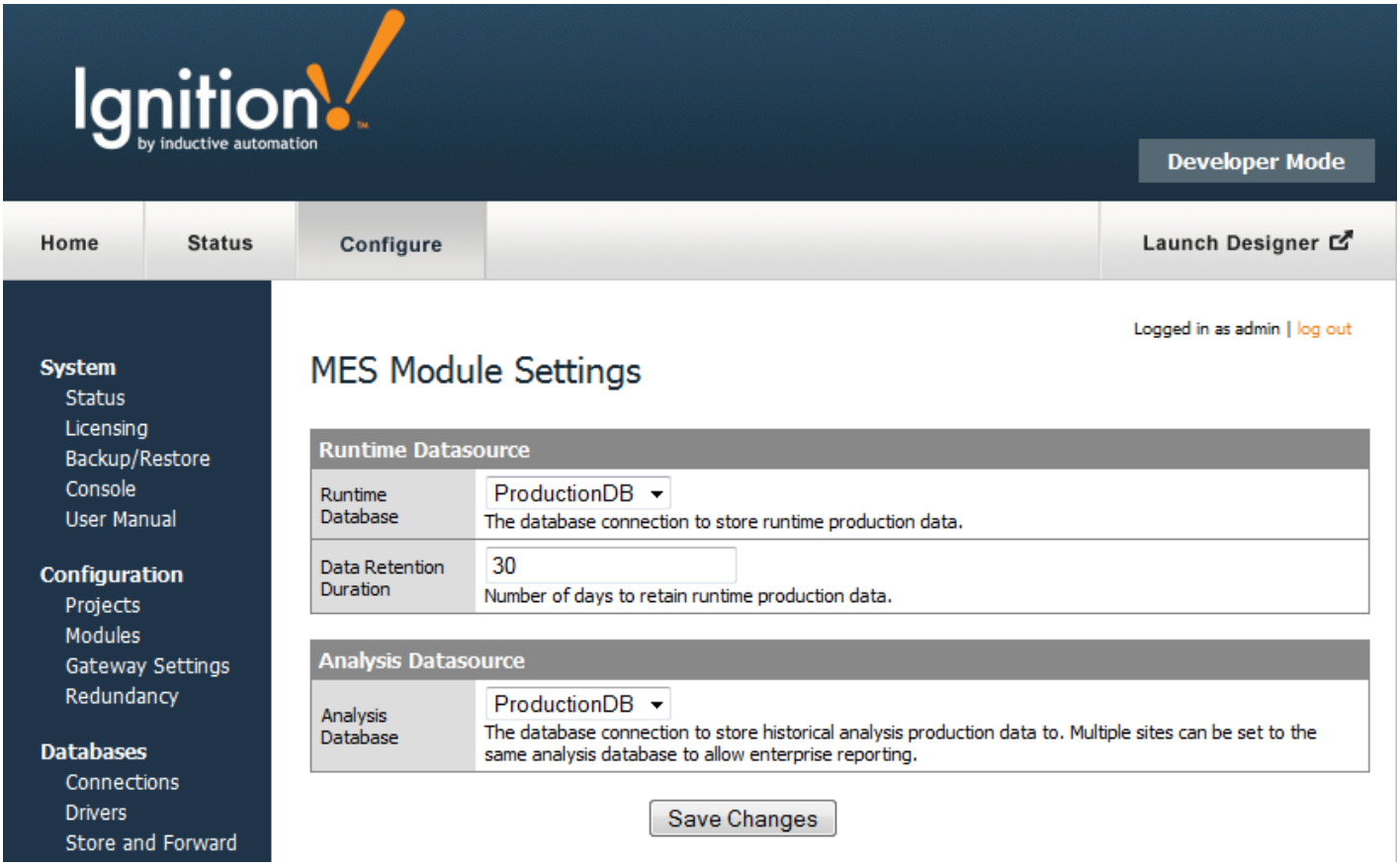
Sample Database Connection

3.2.1.4 MES Module Settings

The OEE Downtime, Scheduling and SPC module stores data in a SQL database. Because Ignition can be configured to multiple databases, the MES Module Settings configuration page is used to select which databases to store OEE, downtime, scheduling and SPC data. If only one database has been configured in Ignition, then it will be selected by default.

To change the MES module settings, go to the configuration section in the gateway and select the **MES Modules> Settings** section from the left-hand side configuration menu.

Once a database connection is created, and if only one database connection exists, then it will be automatically selected to be used by the MES modules.
If more than one database connection exists, then the desired database connection can be selected to be used by the MES modules as shown below.



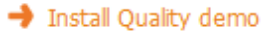
MES Module Settings Page

3.2.1.5 Demo Installation

The QualityDemo project can be used to quickly start using and evaluating the features of the Quality module.

By installing the QualityDemo, SQLTags will be imported, and an SPC Simulator and the demo Ignition project will be installed. To remove the demo, each of these components will have to be manually removed.

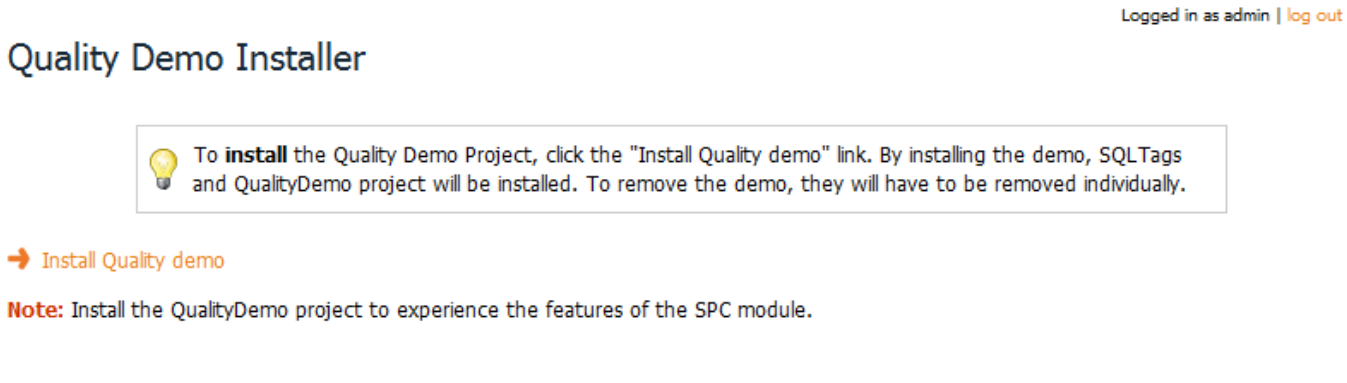
To install the QualityDemo project, go to the configuration section in the gateway and select the **MES Modules> Quality Demo** section from the left-hand side configuration menu. Next, click on the



link.

If the Quality Demo has already been installed, there will be a note stating so. If a database connection has not been installed, a note will appear stating a database connection is needed before installing the demo.

© Inductive Automation


Logged in as admin | log out
Quality Demo Installer
To install the Quality Demo Project, click the "Install Quality demo" link. By installing the demo, SQLTags and QualityDemo project will be installed. To remove the demo, they will have to be removed individually.
Install Quality demo
Note: Install the QualityDemo project to experience the features of the SPC module.

Demo Installation Page

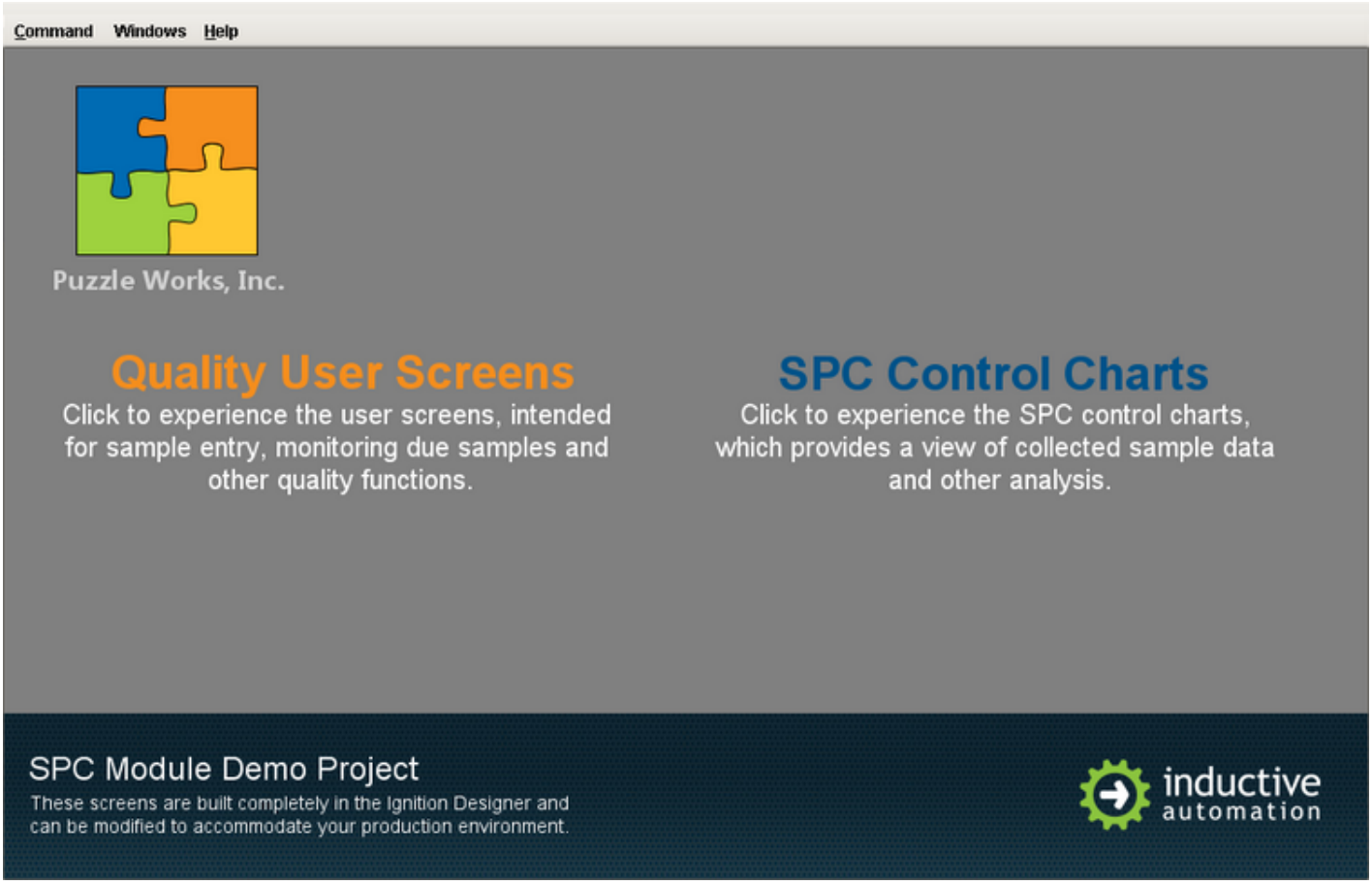
3.2.2 User Interface

This section is a quick walk through of the demo project that is included with the SPC Module. It is intended to provide a starting point for SPC implementation. It can be modified, added to or completely replaced to meet you specific requirements.

The functionality of the SPC demo project includes:

- Sample definitions
- Scheduling samples
- Sample entry
- Late / missing sample indication
- Control charts
- Analysis beyond control charts

The demo is divided into two sections: control charts and user screens. Click on the Quality User Screens or SPC Control Charts for the section you wish to work with.

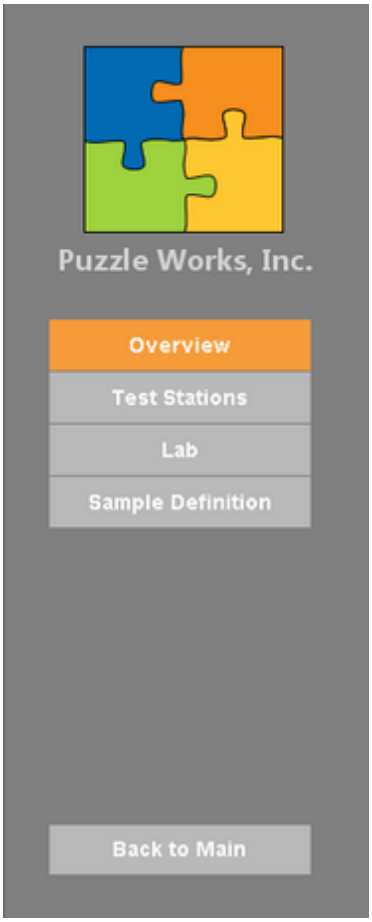

Command Windows Help
Puzzle Works, Inc.
Quality User Screens
Click to experience the user screens, intended for sample entry, monitoring due samples and other quality functions.
SPC Control Charts
Click to experience the SPC control charts, which provides a view of collected sample data and other analysis.
SPC Module Demo Project
These screens are built completely in the Ignition Designer and can be modified to accommodate your production environment.
inductive automation

SPC Module Demo Project Main Screen

3.2.2.1 Quality User Screens

This is the Quality User Screen Menu. Here, the user can click on the menu item to select the corresponding screen or click on the Back to Main to return back to the main demo menu.

© Inductive Automation

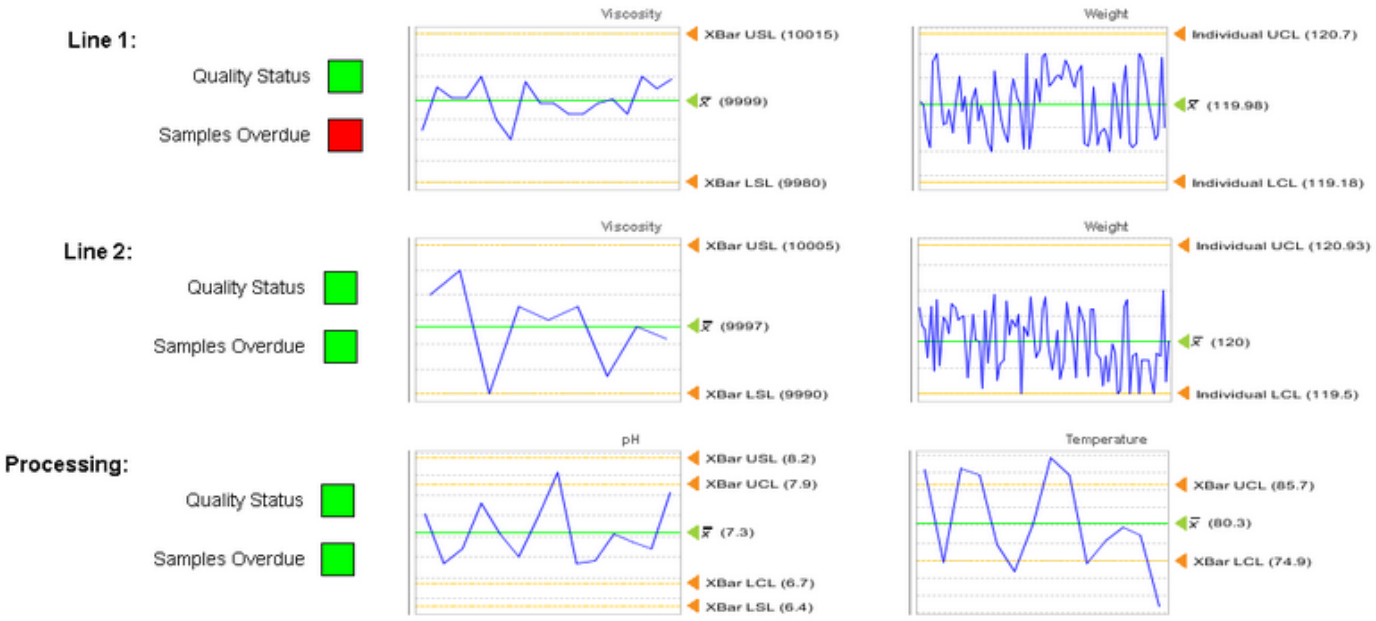


User Screen Menu

3.2.2.1.1 Overview

Below is the overview screen that is included with the SPC demo. It demonstrates control charts that are updated automatically every time new sample measurement data is recorded. It also demonstrates indicators of both overdue samples and processes that are out of control. The indicators can just as easily be alerts that appear in the alarm list or are sent as emails or text messages using the Ignition alerting functionality.

© Inductive Automation



Overview Screen

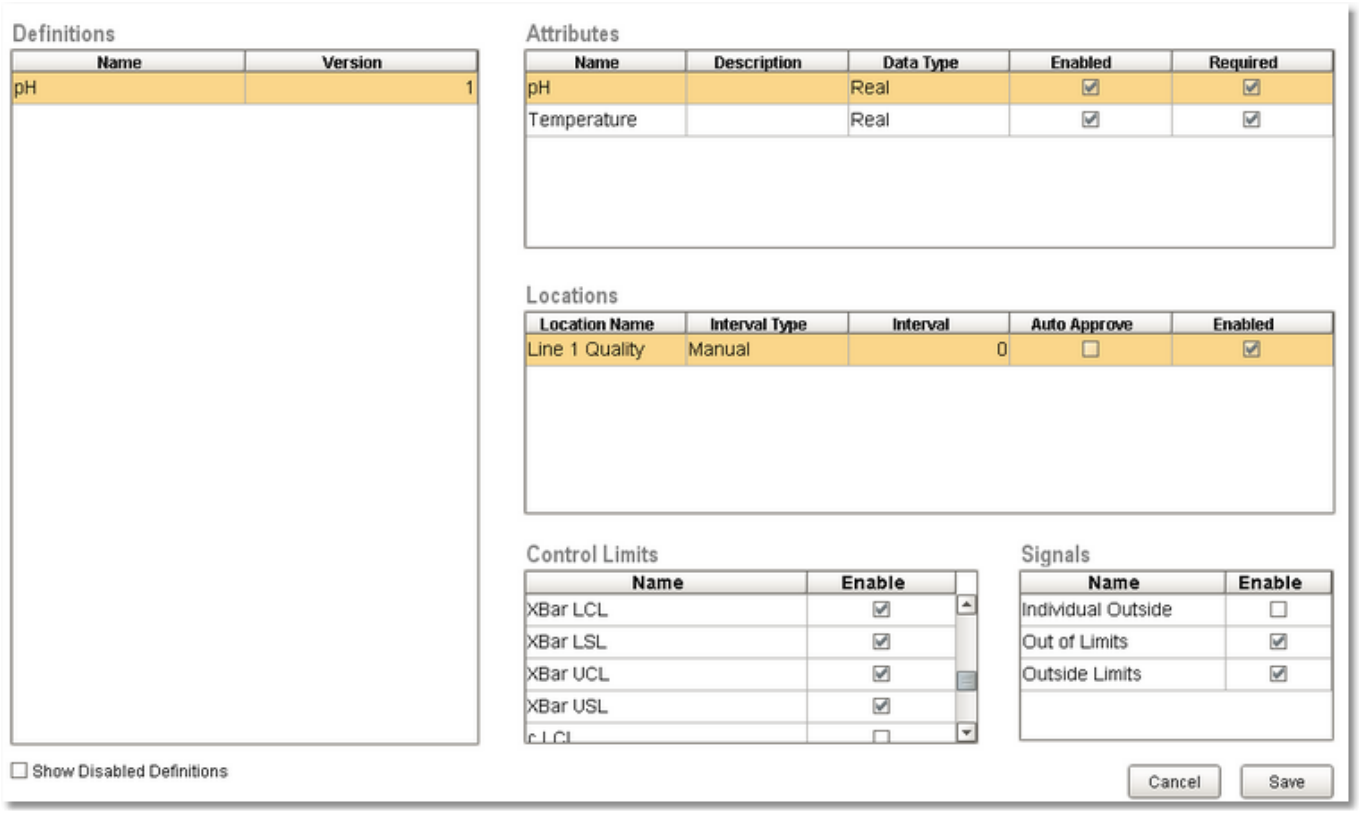
3.2.2.1.2 Sample Definitions

Sample definitions originate from two different sources. One source is the Tag Collectors that are defined in the designer and are for the sole purpose of creating samples automatically from Ignition tags (no human intervention). The other source is from the sample definitions created using the screens covered in this section, and are for the purpose of manual or semi-automatic collection of sample data (human intervention).

The sample definition screen is made up of components from the SPC modules that work together to allow for the management of sample definitions.

By selecting a sample definition, the attributes, locations, control limits and signals associated with it are shown. The attributes define the data measurements to collect for each sample. The locations define the virtual locations that are appropriate for this sample definition. The Control Limits table defines which limits to apply to this sample definition. And last, the signals define which out of control signals to apply to the sample definition.

© Inductive Automation



Sample Definitions Screen

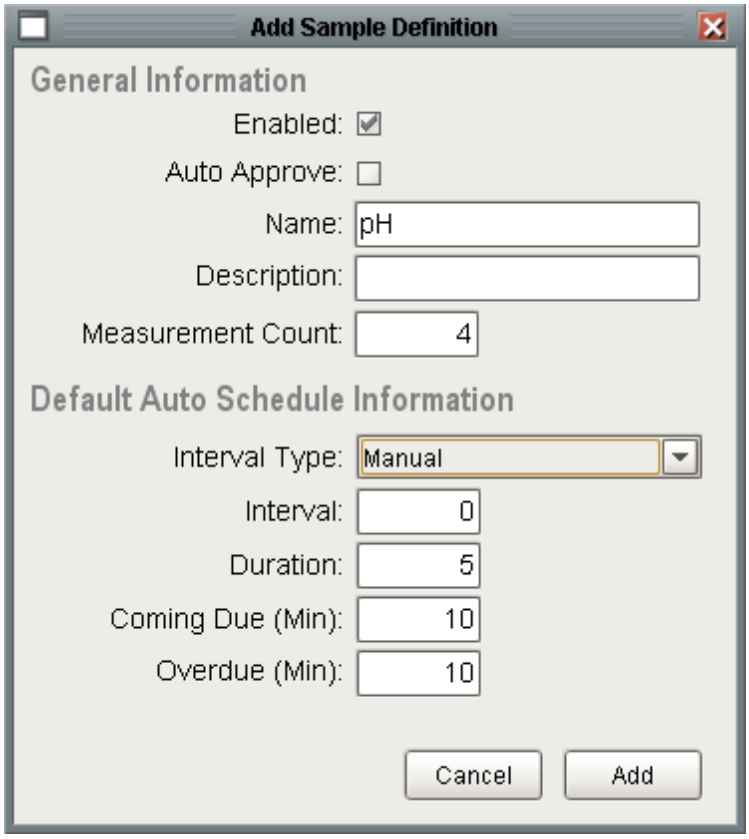
Adding a New Sample Definition

A new sample definition can be added by right-clicking the Definitions Table and selecting "Add" from the drop-down menu. A window will appear, allowing the user to define multiple general information settings.

The auto approve will automatically approve a sample when the measurement data associated with it is recorded. Once a sample is approved, it will appear on the control charts and will included when automatically evaluating for out of control conditions behind the scenes. If the auto approve is not selected then samples based on this sample definition will have to be manually or programmatically approved.

The other general information is straight forward and is described in more detail in the Sample Definition section of this manual. The default auto schedule information defines how samples are scheduled. In the image below, the pH sample definition is set to manual meaning samples are created manually be the user. See the location section below for more information.

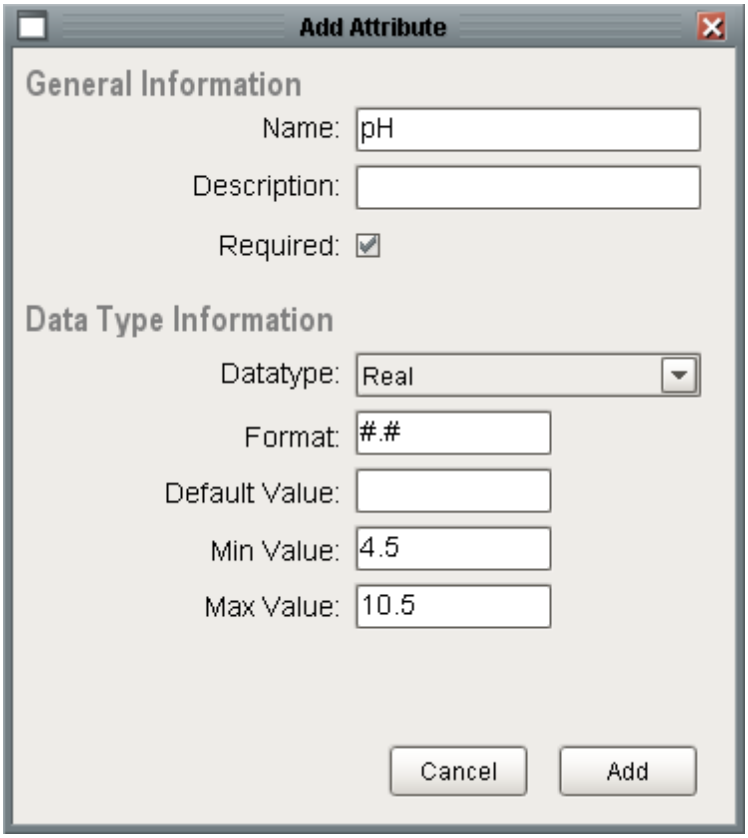
© Inductive Automation

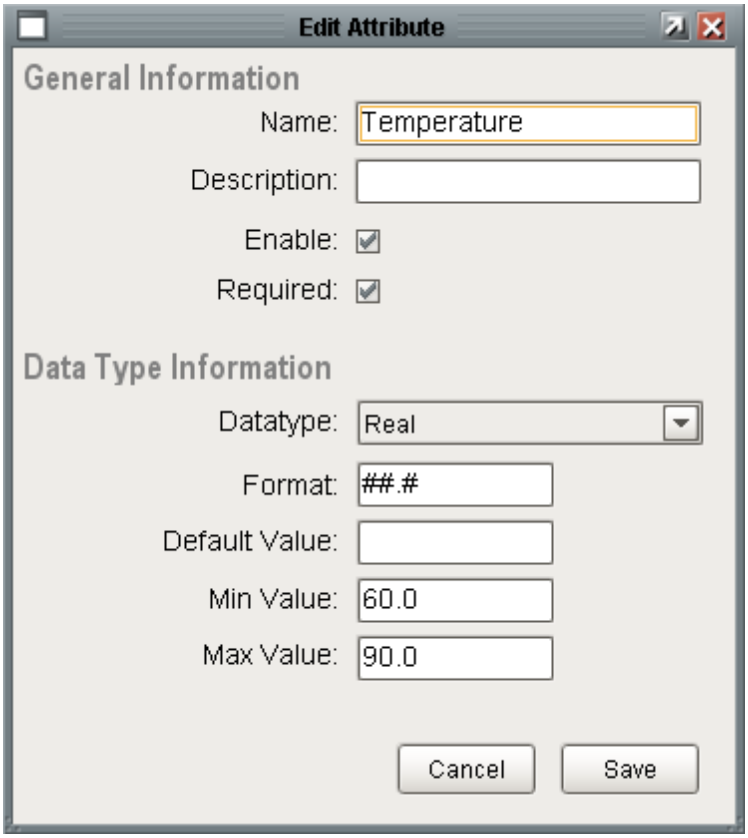


Add Sample Definition Window

After adding a new definition, the attributes must be defined. This is done by right-clicking the Attributes table and selecting "Add" from the drop-down menu. This opens a window similar to the one before, which allows users to define each attribute. Some examples of attributes include pH, temperature, viscosity, weight, nonconformities, and nonconforming items. From here, the name, description, datatype, format, default value, minimum value, and maximum value can be defined. This window also allows the users to decide if the attribute will be required when entering sample data on the Lab or Test Stations screen.

© Inductive Automation



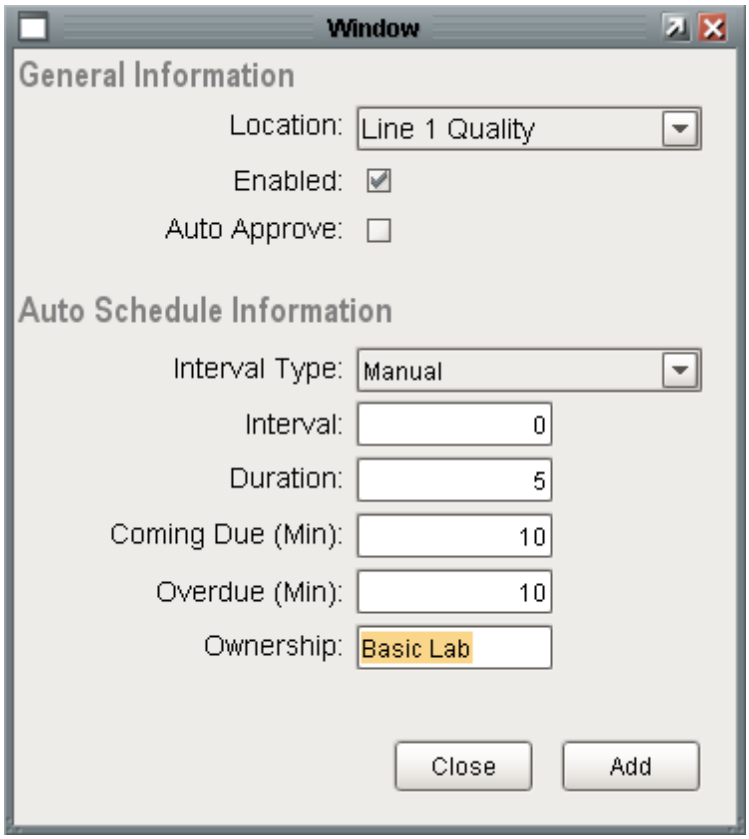


Add and Edit Attribute Windows

© Inductive Automation

Next, the locations, or where the samples will be taken, can be defined. Again, this can be done by right-clicking on the Locations table and selecting "Add" from the drop-down menu. The ownership field declares who is responsible for the testing of the sample, whether that be the lab or the operator at the testing station.

The interval type defines how the samples will automatically be scheduled. Or as in the image below, they will be manually created by the user. If the interval is set to Timed Interval (Hours) then a sample will automatically scheduled as defined by the Interval setting. When a new project is created, the default Intervals options are also created but they can be modified, added to or even removed. See Sample Definition Location for more details of each of the settings.



Add Location Window

Any selected control limits will be available to include on the control charts and will also be included in the automatic evaluation of out of control conditions of the sample data. When a new project is created, the default control limit options are also created but they can be modified, added to or even removed. Keep in mind that each control limit is associated with a particular control chart. For example, XBar UCL is associated and can only be used with the XBar chart. This is because the calculation used to determine the XBar UCL value is specific to only the XBar chart.

© Inductive Automation

Control Limits

Name	Enable
Range UCL	☐
StdDev LCL	☐
StdDev UCL	☐
StdDev XBar LCL	☐
StdDev XBar UCL	☐
XBar LCL	☑
XBar LSL	☑
XBar UCL	☑
XBar USL	☑
c LCL	☐
c UCL	☐
np LCL	☐
np UCL	☐
p LCL	☐
p UCL	☐
u LCL	☐
u UCL	☐

Control Limits Table

Any selected signals will be available to include on the control charts and will also be included in the automatic evaluation of out of control conditions of the sample data. When a new project is created, the default signal options are also created but they can be modified, added to or even removed. Keep in mind that each signal is associated with a particular control chart. For example, Individual Outside is associated with, and can only be used with, the Individual chart. This is because the calculation and control limits used to determine if a sequence of individual values are out of control is specific to the Individual chart.

Signals

Name	Enable
Individual Outside	☐
Out of Limits	☑
Outside Limits	☑

Signals Table

After all the desired settings have been defined, the user can select "Save" to commit all the changes, or "Cancel" to undo any changes that have been made.

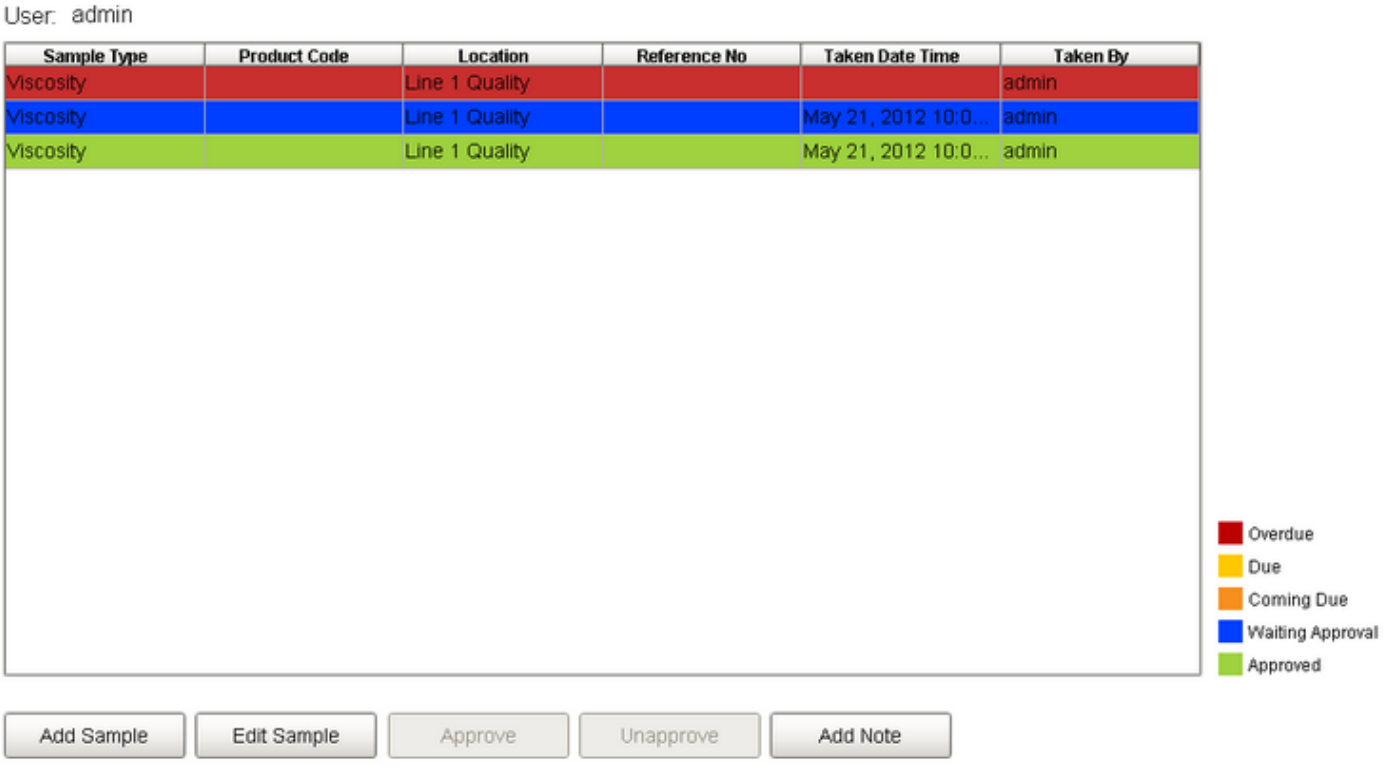
© Inductive Automation

After a sample definition has been created, samples based on them may appear or be manually added depending on the Interval setting.

3.2.2.1.3 Sample Entry

Although it is not required, the sample list is used to view samples coming due, due, overdue, waiting for approval and approved. Based on the color, users can easily see the current state of samples.

From this list, users can select a sample to enter measurements for or create new samples. See the sample definition section for more information about how to schedule samples or define them to be taken manually. By selecting a sample and clicking on the Edit Sample button, the sample data can be entered. Likewise, by clicking on the Add Sample button, a new sample can be added. Depending on the sample definition, samples can be automatically or manually approved. Once a sample has been approved, it will appear in the control charts and will be automatically evaluated for an out of control condition. In this demo, the Unapprove Sample button has been added to demonstrate the ability to correct previously approved sample data. This can be removed from the screen or allowed based on the user's security role.



Sample List

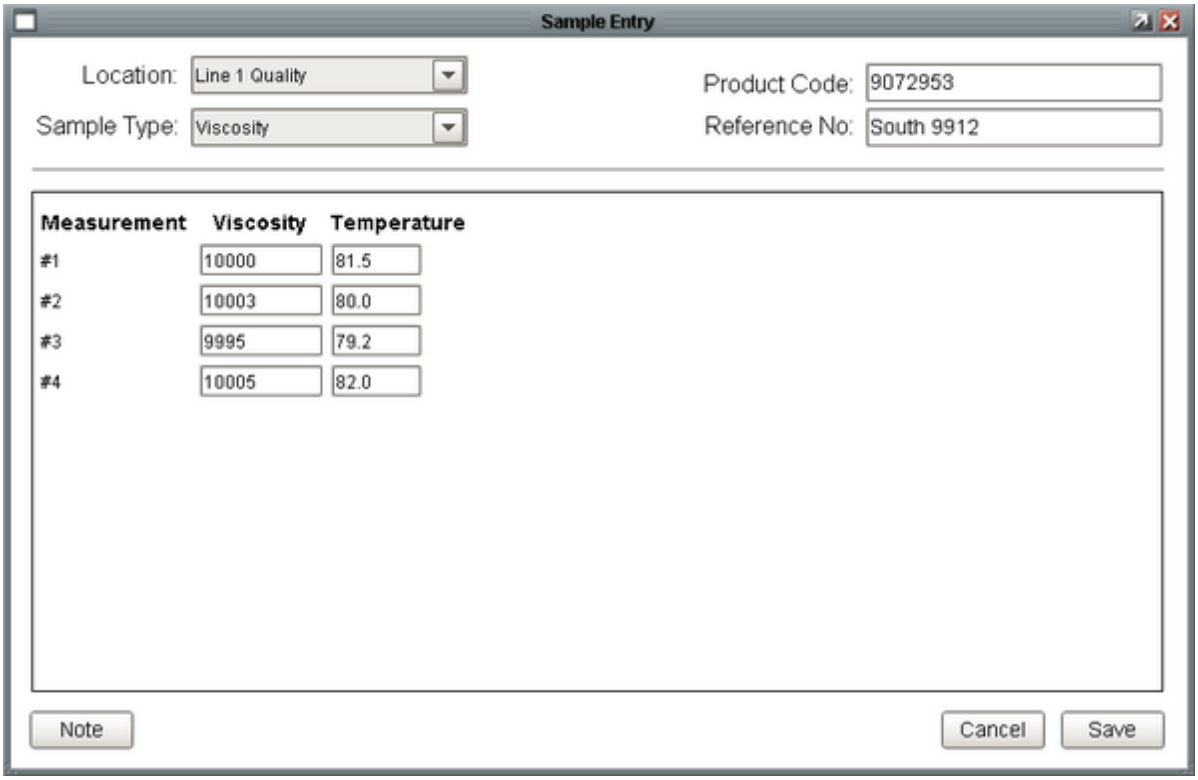
Once the user has clicked on the Edit Sample or Add Sample button, the sample entry form appears. If a new sample has been added, the location can be selected. For a location to appear as an option here, it must first be added to the location list for the desired sample definition with the Ownership setting set to "Lab" for the lab entry screen or "Test Station" test station entry screen. These ownership tags can be changed using the designer or additional ownership tags can be added.

The following screen shows the entering of measurements for a value based sample. In this case,

© Inductive Automation

viscosity and temperature values. Users also have the ability to enter a product code and reference number (located in the upper right-hand corner). These can be used when viewing the samples in the control charts or for analysis beyond control charts.

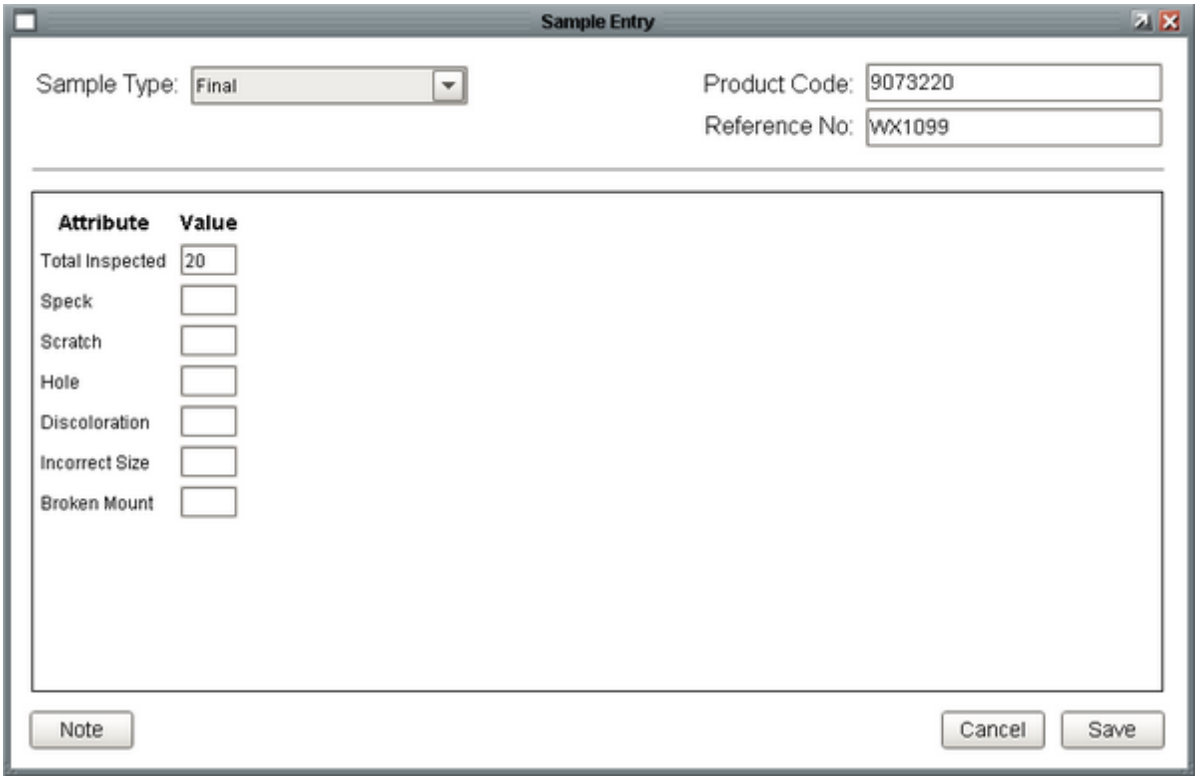
Because multiple measurements are being entered for each attribute, the attributes appear horizontally and the measurements vertically. If the sample definition only calls for one measurement, then the attributes will appear vertically.



Value Sample Entry

Below represents entering data for a attribute based sample.

© Inductive Automation



Attribute Sample Entry

3.2.2.2 SPC Control Charts

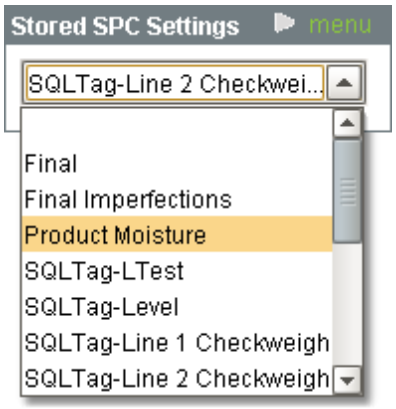
This is the SPC Control Chart Menu. Here, the user can click on the menu item to select the corresponding control chart or click on the Back to Main to return back to the main demo menu.

© Inductive Automation

Control Chart Menu

3.2.2.2.1 Control Charts

When a sample definition is created, it will appear as an option in the Stored SPC Settings selection box.



Stored SPC Settings Selection Box

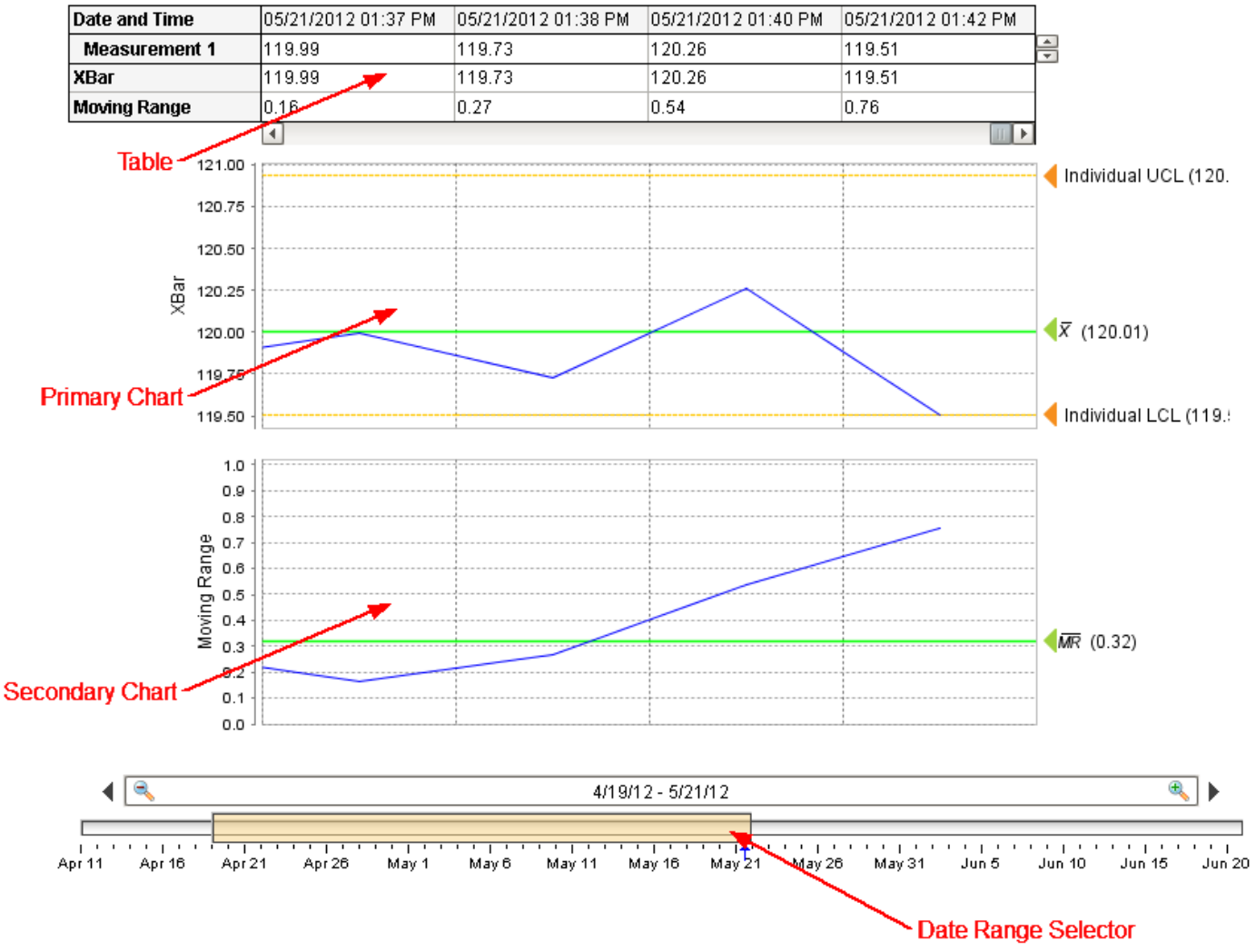
© Inductive Automation

Note: When the demo is first installed there is no SPC data. After it has run for 10 minutes or so, the SQLTag-Line 1 Checkweigher and SQLTag-Line 2 Checkweigher options will have collected a sample amount of data. If any samples have been entered on the sample entry screens, they will appear as well.

After selecting one of the Stored SPC Settings options, the appropriate control chart will be shown. From here other options can be selected, which will be discussed later on.

The image below labels the major parts of the control chart. The Date Range Selector is used to select the date range of samples to view. It defaults to the current period of time, but can be used to select samples from the past. The table shows the data collected and the calculated values. The calculated values that are included depends on the kind of control chart being displayed. When the scroll bar at the bottom of the table is moved to the left, the table, primary chart and secondary chart will all scroll in unison to previous samples within the selected date range.

For the attribute type of control charts the secondary table will not appear.



Control Chart

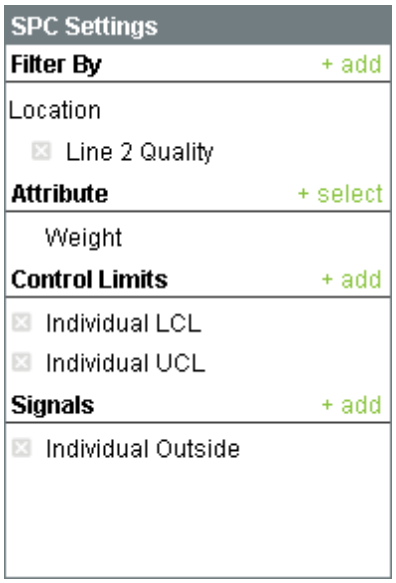
Changing what attribute is currently being shown in the control chart is done using the SPC settings panel. To

© Inductive Automation

change the attribute, click on the ***+ select*** to the right of the Attribute label. This will show all of the attributes defined in the sample definition. In the case of the SQLTag-Line 1 Checkweigher, only one attribute is available.

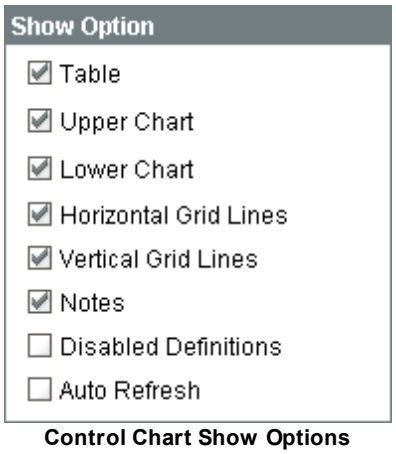
Control limits and signals can be selected or hidden using the same method as the attribute with the exception that more than one control limit or signal can be selected.

The filter by section allows for the limiting of samples that will be shown and included in the calculated values. At a minimum, at least one location must be specified. This is because data collected from one location could be completely unrelated or in a different range than another location. If this is not the case, then multiple locations can be added to the filter.



SPC Settings

The show options allow for the appearance of the control chart to be changed. By removing the Table option, the table will not appear leaving only the charts and allowing more samples to be viewed at once.

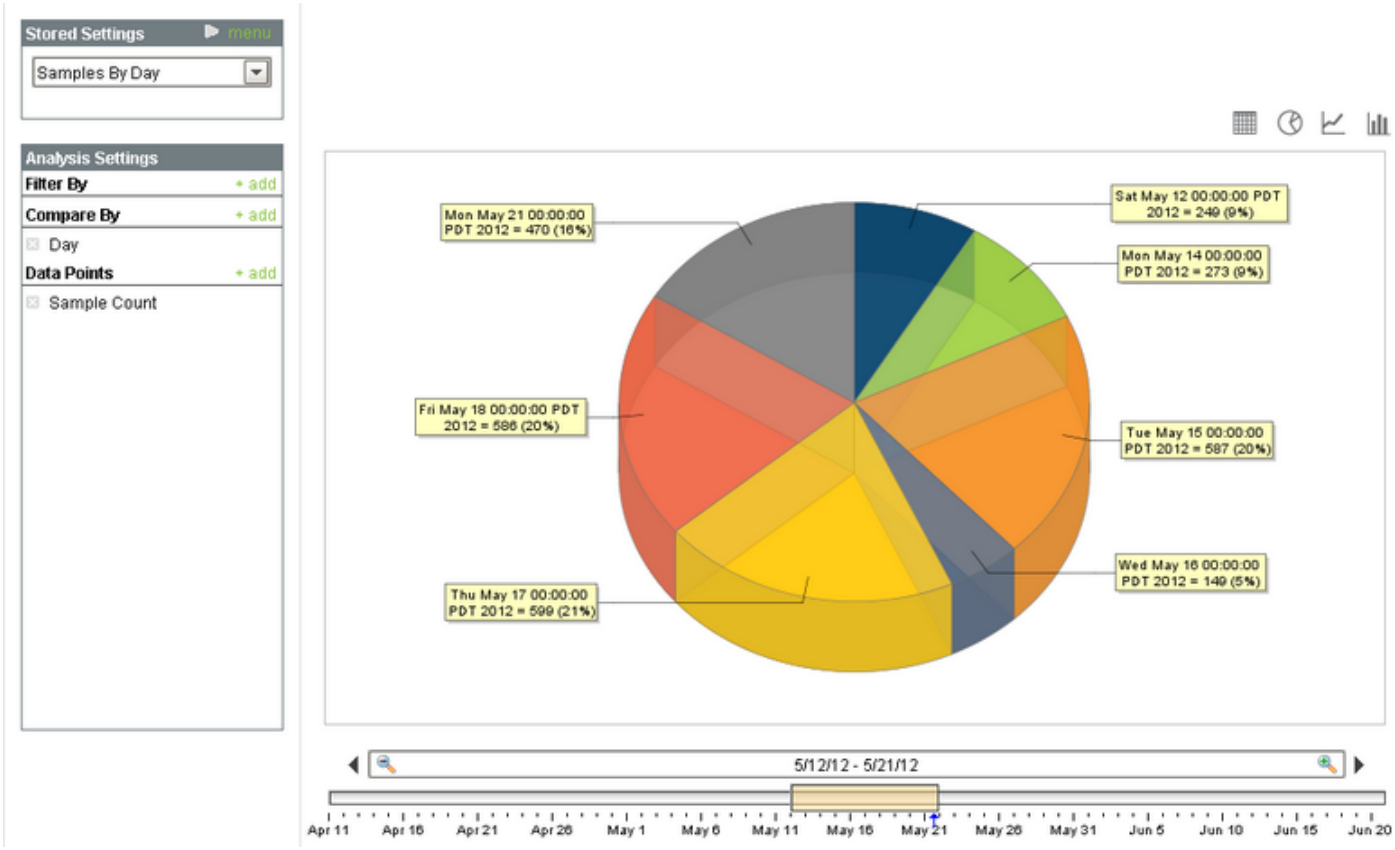


Control Chart Show Options

© Inductive Automation

3.2.2.2.2 Analysis

The analysis screen allows for free form analysis of production and quality data. This data can also be filtered to include only specific criteria. Additionally, comparisons can be made between different factors. For example, sample count by operator can be analyzed, or even process out of control conditions by operator by shift. The four icons in the upper right corner are used to select between pie chart, bar chart, line chart or tabular format.

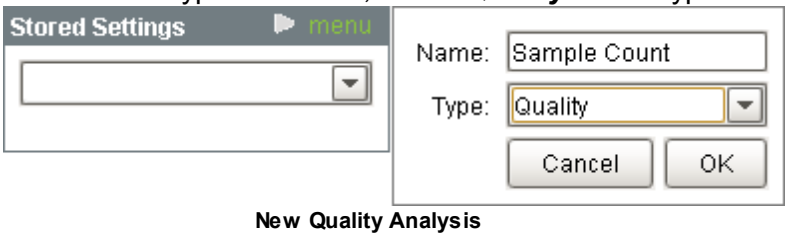


SPC Analysis Screen

The date range selector at the bottom is used to define the data range that will be included in the analysis. As you change the start or end dates, only the production runs that are within that range will be included in the analysis.

Stored Analysis

Start out by creating a new analysis by clicking on the ***menu*** of the Stored Settings panel and then selecting the ***New*** menu item. Next type in a name, select ***Quality*** for the type and click the OK button.

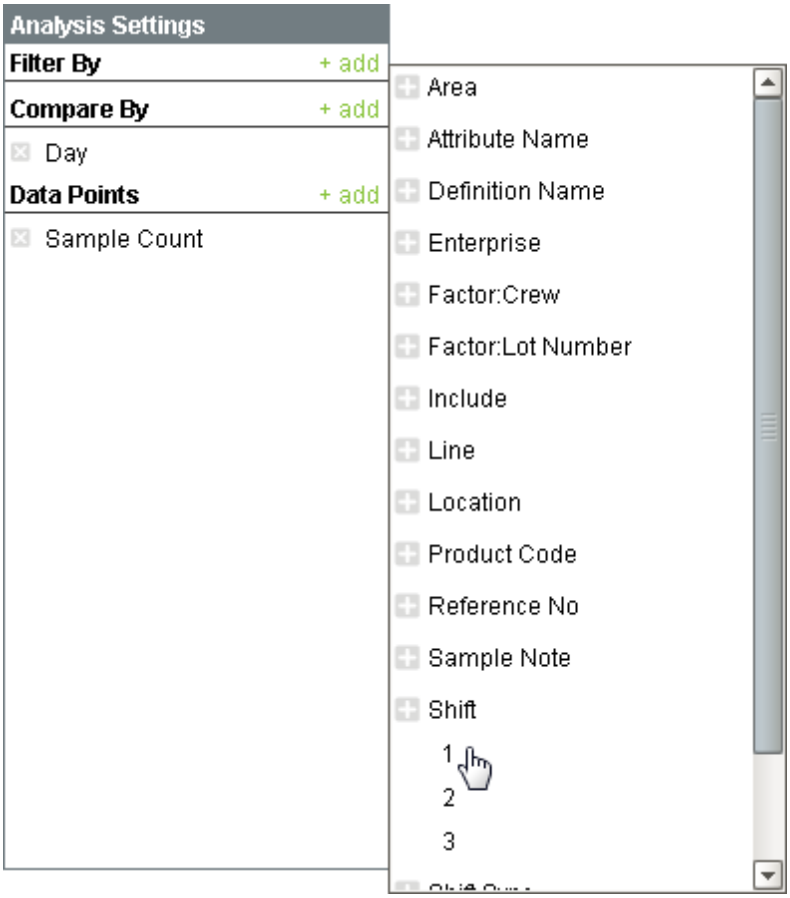


New Quality Analysis

Filter By

Once an stored analysis has been created or selected, you can change the selections to zero in on the

data that is desired. The filter section allows you to limit the data that is included in the analysis. Filters can be added by clicking on the + add icon on the right side of the *Filter By* section. Within the popup filter selection window, scroll down to the **Shift** option and click the icon. Notice the shifts can now be selected. Clicking on 1 for first shift will add the Shift = 1 causing the analysis results to included quality data for only for first shift. Any combination on filters can be added and the corresponding results will be shown.



Filter By Options

The list of available filters change based on the date range. For example, if no samples were taken during the second shift, then a 2 will not appear as an available option under shift.

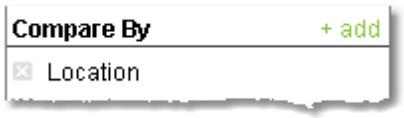
Filter By items can be removed by clicking on the located to the left of the filter name.

Compare By

Breaking up information into groups is more meaningful than just seeing a total for a given date range. For example, knowing the total sample count for a given data range does not provide actionable information that can be used to improve quality. Now, comparing by the sample count for each person entering sample data may provide meaningful and actionable data that can be used to determine staffing requirements.

Additional Compare By items can be added by clicking on the + add icon on the right side of the *Compare By* section. Within the popup Compare By selection window, click on the item that you want to compare analysis results between.

© Inductive Automation

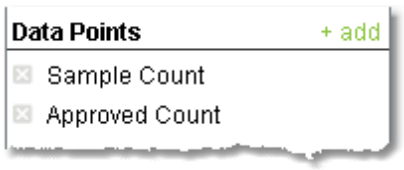


Compare By Selections

Compare By items can be removed by clicking on the ⊠ located to the left of the name.

Data Points

Data points are the individual pieces of information that will be present in the analysis. For example, sample count or approved count are just two of the many available data points. To add a data point, click on the + add icon on the right side of the *Data Points* section. Within the popup Data Point selection window, click on the data point item to include in the analysis.



Data Point Selections

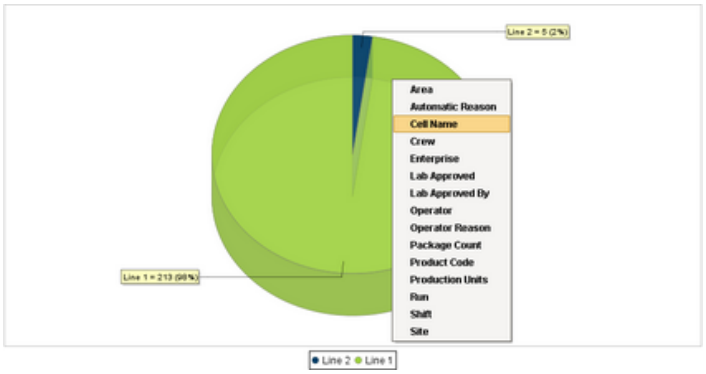
Data Points can be removed by clicking on the ⊠ located to the left of the name.

The pie chart will only show one data point. For this reason if more than one data point is selected the bar chart, line chart or table must be selected to see all the selected data points.

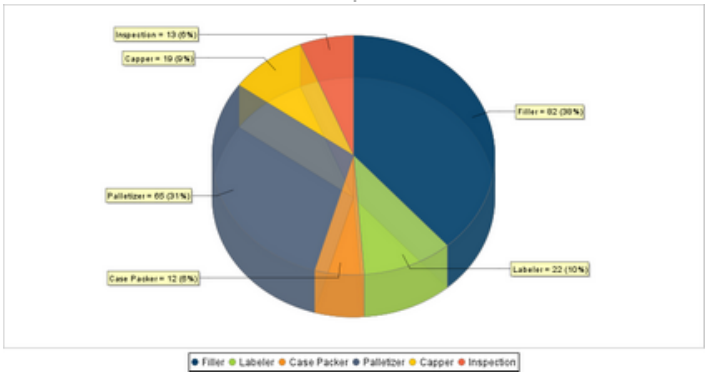
© Inductive Automation

Drill Down

The drill down feature simplifies the compare by and filter selections. Click on a chart series to display the available drill down options. As shown in **Drill Down Example 1** below, clicking on the Line 1 Quality pie segment will show a popup menu of drill down options. If the **Shift** option is selected, then the analysis filters will show the information by Shift and the Filter By and the Compare By sections add *Shift*. The result is shown in **Drill Down Example 2**. Again, by clicking on the pie segment and selecting another drill down option, the Filter By and Compare By selections will change to show the appropriate information . This can be continued any number of times.



Drill Down Example 1



Drill Down Example 2

3.3 Configuration

There are two areas to configure the SPC Quality module. The first area is in the Ignition Gateway and affects all SPC Modules.

The second is in the Ignition Designer and is used to configure production models, user screens and the like. These settings are saved in an Ignition project and can be backed up and restored using the built-in project backup and restore features of Ignition.

3.3.1 MES Module Configuration

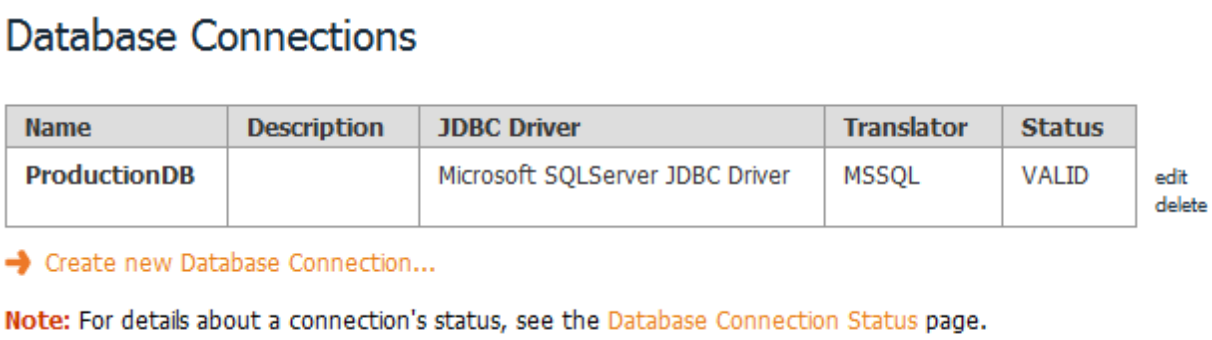
The SPC Quality module is just one of the SPC (Statistical Process Control) modules that has settings which can be set.

© Inductive Automation

3.3.1.1 Datasource Settings

OEE, downtime and schedule data is stored in databases external to Ignition. These database(s) are setup in the gateway configuration section by selecting the **Databases> Connections** section from the left-hand configuration menu in Ignition. See the Ignition documentation for more information on setting up a database connection.

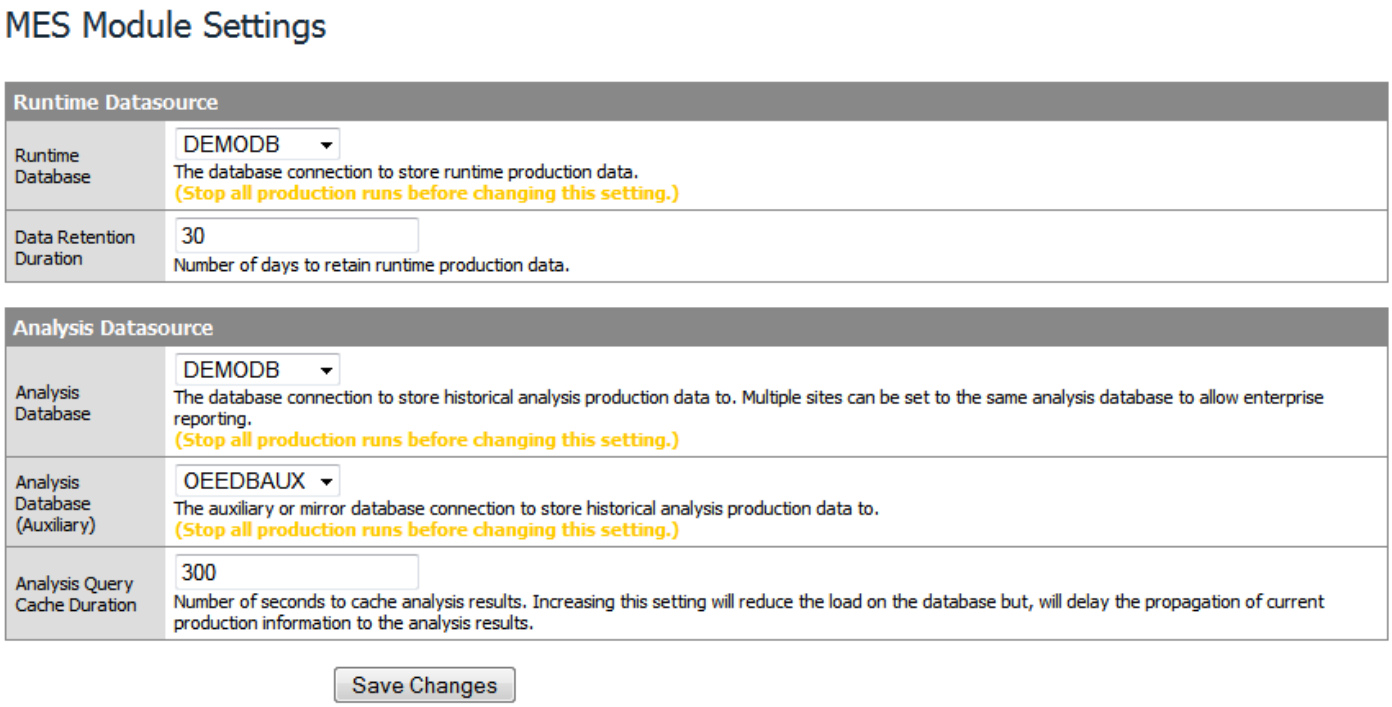
Below shows a typical database connection that is required for the OEE, Downtime and Scheduling module.



Sample Database Connection

To change the MES module settings, go to the configuration section in the gateway and select the **MES Modules> Settings** section from the left-hand side configuration menu.

Once a database connection is created, and if only one database connection exists, then it will be automatically selected to be used by the MES modules.
If more than one database connection exists, then the desired database connection can be selected to be used by the MES modules as shown below.


MES Module Settings
Runtime Datasource
Runtime Database
DEMODB
The database connection to store runtime production data.
(Stop all production runs before changing this setting.)
Data Retention Duration
30
Number of days to retain runtime production data.
Analysis Datasource
Analysis Database
DEMODB
The database connection to store historical analysis production data to. Multiple sites can be set to the same analysis database to allow enterprise reporting.
(Stop all production runs before changing this setting.)
Analysis Database (Auxiliary)
OEEDBAUX
The auxiliary or mirror database connection to store historical analysis production data to.
(Stop all production runs before changing this setting.)
Analysis Query Cache Duration
300
Number of seconds to cache analysis results. Increasing this setting will reduce the load on the database but, will delay the propagation of current production information to the analysis results.
Save Changes

© Inductive Automation

Runtime Database
The runtime database is where production and downtime data is stored during a production run. During a production run, data is logged every minute or partial minute if a downtime event occurs, so a larger amount of data is stored in the runtime database.

Data Retention Duration
This setting specifies the number of days to retain the data in the runtime database after a production run has completed. The default setting is 30 days, This allows for viewing current and past production run information, down to the minute, for the past 30 days.

Analysis Database
The analysis database is where summarized production and downtime data is saved. For single production *site* installations, this can be the set to the same database as the runtime database. For multi-production *site* installations, all *sites* must set the analysis database to the same database to allow for *enterprise* analysis and reporting.

Analysis Database (Auxiliary)
The MES Modules will mirror the hsitorical analysis data that is written to the local analysis database to this database. For single site implementations, set this to "-none-". For multi-site implementations, set this to the datasource for the common remote enterprise database.

Analysis Query Cache Duration
This setting represents the number of seconds to cache analysis results.

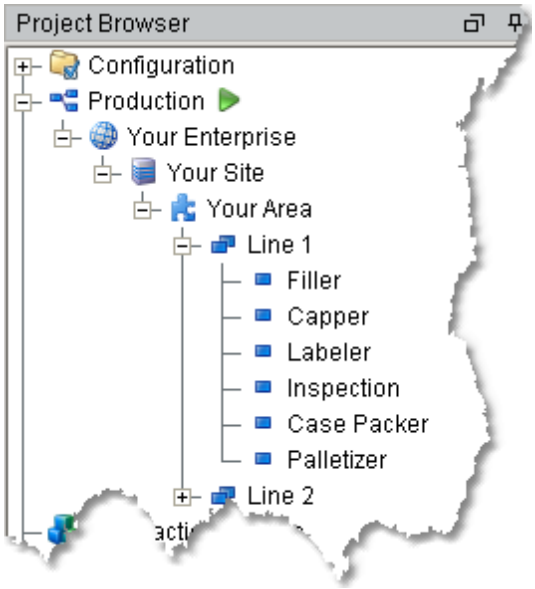
3.3.2 Production Model Configuration

A production model defines your manufacturing or process in tree view form. It provides an organized way to easily configure, control and analyze your facility. It starts with your *enterprise*, which represents your company, and continues down to the *site* (physical location), *area, location*, *line* and *cells*.

3.3.2.1 Production Module

The production model is configured within the Ignition designer and is accessed by selecting the "Production" folder in the project browser. From here, your *enterprise*, *site*, *area(s)*, *line(s)* and *cell(s)* can be added, renamed and deleted.

© Inductive Automation



Production Model Tree

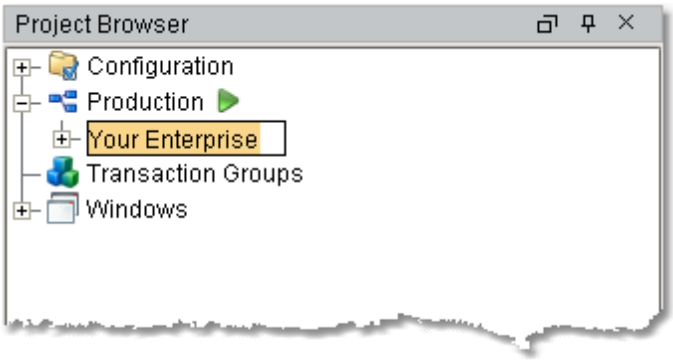
3.3.2.1.1 Enterprise Configuration

Adding an Enterprise

To add your *enterprise*, right-click on the "Production" folder in the project browser and select the **New Production Item > New Production Enterprise** menu item. An *enterprise* named "New Enterprise" will be added to the "Production" folder.

Renaming an Enterprise

To rename it to the name of your *enterprise*, right-click on it and select **Rename**, then enter the new name.



Enterprise Name

Deleting an Enterprise

To remove an existing *enterprise*, right-click on the *enterprise* item and select the **Delete** menu item. A window will appear confirming that you permanently want to delete the production *enterprise*. Please note that the *site*, *area(s)*, *line(s)* and *cell(s)* underneath the *enterprise* will also be permanently removed.

© *Inductive Automation*

General Enterprise Settings

These settings are accessed by selecting the *enterprise* item contained in the ***"Production"*** folder in the project browser and then selecting the "General" tab as shown below.

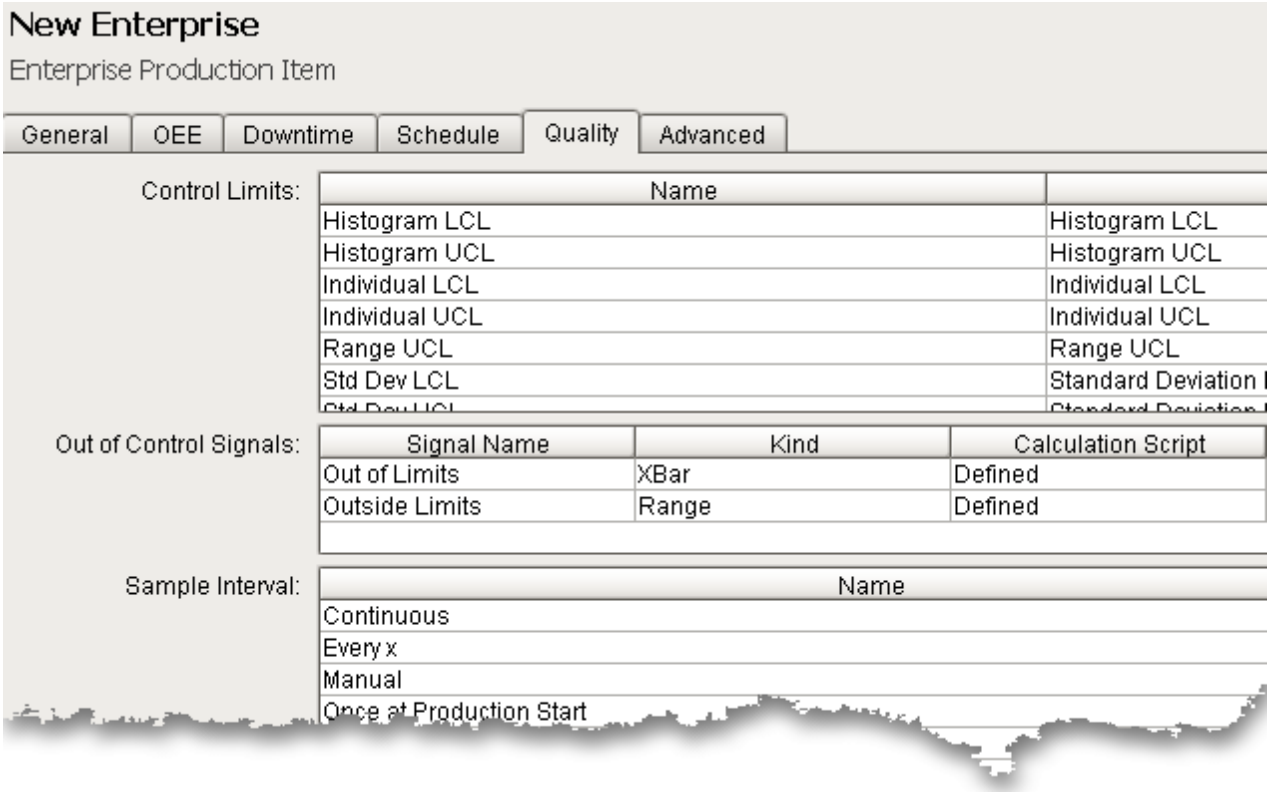
Enterprise General Settings

Enabled By default, added *enterprises* are enabled. It can be disabled by un-checking the Enabled setting and saving the project. This will stop the OEE, downtime and scheduling module from executing the *enterprise*, the *site* and all *area(s)*, *line(s)* and *cell(s)* that are underneath it.

Description This is an optional description and is just for your reference.

Enterprise Quality Settings

These settings are accessed by selecting the *enterprise* item contained in the *"Production"* folder in the project browser and then selecting the "Quality" tab as shown below.



Enterprise Quality Settings

From here, the Control Limits, Out of Control Signals, and Sample Intervals can be defined. This definition will determine what options will appear for every sample that is defined. For example, if the Histogram LCL control limit is not defined on the Enterprise page, it will not be an available option when selecting control limits on the Sample Definition page. Default control limits, out of control signals, and sample intervals will be present and these may be edited or deleted. New limits, signals or intervals can also be added.

Adding a Control Limit, Out of Control Signal, or Sample Interval

To add a limit, signal, or interval, right-click on the table and select New. After filling in the necessary fields, select OK.

Editing a Control Limit, Out of Control Signal, or Sample Interval

To edit a limit, signal, or interval, select the item to be edited, right-click, and select Edit from the drop-down menu. After making the desired changes, select OK.

Deleting a Control Limit, Out of Control Signal, or Sample Interval

To delete a limit, signal, or interval, select the item to be deleted, right-click, and select Delete from the drop-down menu.

For more information, view the pages on Control Limits, Out of Control Signals, and Sample Intervals.

3.3.2.1.2 Site Configuration

Adding a Site

To add your *site*, right-click on your *enterprise* folder in the project browser and select the **New Production Item > New Production Site** menu item. A *site* named "New Site" will be added to the *enterprise* folder.

© Inductive Automation

Renaming a Site

To rename it to the name representing the *site's* physical location, right-click on it and select **Rename**, then enter the new name.

Deleting a Site

To remove an existing *site*, right-click on the *site* item and select the **Delete** menu item. A window will appear confirming that you permanently want to delete the production *site*. Please note that the *area(s)*, *line(s)* and cell(s) underneath the *site* will also be permanently removed.

New Site

General Site Settings

These settings are accessed by selecting the *site* item contained in the *enterprise* folder in the project browser, and then selecting the "General" tab.

Enabled — By default, added *sites* are enabled. It can be disabled by un-checking the Enabled setting and saving the project. This will stop the OEE, downtime and scheduling module from executing the *site* and all *area(s)*, *line(s)* and *cell(s)* that are underneath it.

Description — This is an optional description and is just for your reference.

Shift 1:

- **Default Enabled** — If checked, shift 1 will be included during scheduling. If not checked, shift 1 will be scheduled around.
- **Default Start Time** — The time of day that first shift starts. The first shift ends at the start of second shift.

Shift 2:

- **Default Enabled** — If checked, shift 2 will be included during scheduling. If not checked, shift 2 will be scheduled around.
- **Default Start Time** — The time of day that second shift starts. The second shift ends at the start of third shift.

Shift 3:

- **Default Enabled** — If checked, shift 3 will be included during scheduling. If not checked, shift 3 will be scheduled around.
- **Default Start Time** — The time of day that third shift starts. The third shift ends at the start of first shift.

Note: The shift enabled and shift start times are the default for your production site and can be overridden by the production area and/or production line.

© Inductive Automation

3.3.2.1.3 Area Configuration

Adding an Area

To add a production *area*, right-click on your *site* folder in the project browser and select the **New Production Item > New Production Area** menu item. An *area* named "New Area" will be added to the *site* folder. Multiple production *areas* can be added to your production *site*. Each *area* can represent a physical or logical production *area* within your production *site*. Some examples of production *areas* are: packaging, cracking, filtration, fabrication, etc.

Renaming an Area

To rename it to the name representing the production *area*, right-click on it and select **Rename**, then enter the new name.

Deleting an Area

To remove an existing production *area*, right-click on the *area* item and select the **Delete** menu item. A window will appear confirming that you permanently want to delete the production *area*. Please note that the *line(s)* and *cell(s)* underneath the *area* will also be permanently removed.

New Area

Area General Settings

These settings are accessed by selecting the desired *area* item contained in the *site* folder in the project browser and then selecting the "General" tab.

Enabled — By default, added *areas* are enabled. It can be disabled by un-checking the Enabled setting and saving the project. This will stop the OEE, downtime and scheduling module from executing the *area* and all *line(s)* and *cell(s)* that are underneath it.

Description — This is an optional description and is just for your reference.

Shift 1

- **Default Enabled** — If checked, shift 1 will be included during scheduling. If not checked, shift 1 will be scheduled around. To inherit the shift enabled from the from the *site*, select the "Inherit From Parent" option.
- **Default Start Time** — The time of day that first shift starts. The first shift ends at the start of second shift. To inherit the time of day that first shift starts setting from the *site*, select the "Inherit From Parent" option.

Shift 2

- **Default Enabled** — If checked, shift 2 will be included during scheduling. If not checked, shift 2 will be scheduled around. To inherit the shift enabled from the from the *site*, select the "Inherit From Parent" option.
- **Default Start Time** — The time of day that second shift starts. The second shift ends at the start of third shift. To inherit the time of day that second shift starts setting from the *site*, select the "Inherit From Parent" option.

© Inductive Automation

Shift 3

Default Enabled If checked, shift 3 will be included during scheduling. If not checked, shift 3 will be scheduled around. To inherit the shift enabled from the from the *site*, select the "Inherit From Parent" option.

Default Start Time The time of day that third shift starts. The third shift ends at the start of first shift. To inherit the time of day that third shift starts setting from the *site*, select the "Inherit From Parent" option.

Note: The shift start times are the default for your production site and can be overridden by the production area and/or production line.

3.3.2.1.4 Line Configuration

Adding a Line

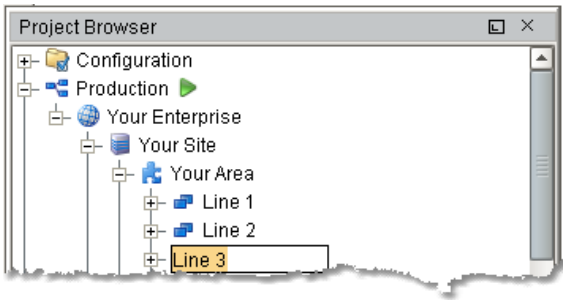
To add a production *line*, right-click on an *area* folder in the project browser and select the **New Production Item > New Production Line** menu item. A *line* named "New Line" will be added to the *area* folder. Multiple production *lines* can be added to a production *area*.

Renaming a Line

To rename it to the name representing the production *line*, right-click on it and select **Rename**, then enter the new name.

Deleting a Line

To remove an existing production *line*, right-click on the *line* item and select the **Delete** menu item. A window will appear confirming that you permanently want to delete the production *line*. Please note that the cell(s) underneath the *line* will also be permanently removed.



New Line

Line General Settings

These settings are accessed by selecting the desired *line* item contained in the *area* folder in the project browser and then selecting the "General" tab.

Enabled By default, added *lines* are enabled. It can be disabled by un-checking the Enabled setting and saving the project. This will stop the OEE, downtime and scheduling module from executing the *line* and *cell(s)* that are underneath it.

Description This is an optional description and is just for your reference.

Shift 1

Default Enabled If checked, shift 1 will be included during scheduling. If not checked, shift 1 will be scheduled around. To inherit the shift enabled from the from the *area*, select the "Inherit From Parent" option.

© Inductive Automation

Default Start Time The time of day that first shift starts. The first shift ends at the start of second shift. To inherit the time of day that first shift starts setting from the *area*, select the "Inherit From Parent" option.

Shift 2

Default Enabled If checked, shift 2 will be included during scheduling. If not checked, shift 2 will be scheduled around. To inherit the shift enabled from the from the *area*, select the "Inherit From Parent" option.

Default Start Time The time of day that second shift starts. The second shift ends at the start of third shift. To inherit the time of day that second shift starts setting from the *area*, select the "Inherit From Parent" option.

Shift 3

Default Enabled If checked, shift 3 will be included during scheduling. If not checked, shift 3 will be scheduled around. To inherit the shift enabled from the from the *area*, select the "Inherit From Parent" option.

Default Start Time The time of day that third shift starts. The third shift ends at the start of first shift. To inherit the time of day that third shift starts setting from the *area*, select the "Inherit From Parent" option.

3.3.2.1.5 Location Configuration

Adding a Location

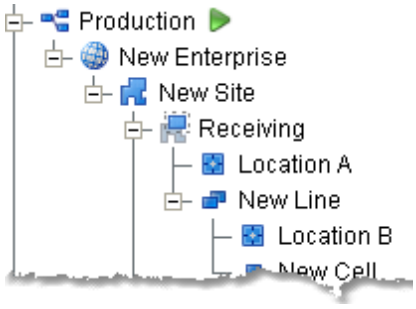
To add a production *location*, right-click on an *area* or *line* folder in the project browser and select the **New Production Item > New Production Location** menu item. A *location* named "New Location" will be added to the *area* or *line* folder. Multiple production *locations* can be added to a production *area* or *line*.

Renaming a Location

To rename it to the name representing the production *location*, right-click on it and select **Rename**, then enter the new name.

Deleting a Location

To remove an existing production *location*, right-click on the *location* item and select the **Delete** menu item. A window will appear confirming that you permanently want to delete the production *location*. Please note that the line(s) and cell(s) underneath the *location* will also be permanently removed.



New Location

Location General Settings

These settings are accessed by selecting the desired *location* item contained in the *area* or *line* folder in the project browser and then selecting the "General" tab.

Enabled By default, added *lines* are enabled. It can be disabled by un-checking the Enabled setting and saving the project. This will stop the OEE, downtime and scheduling module from executing the *line* and *cell(s)* that are underneath it.

Description This is an optional description and is just for your reference.

Shift 1

Default Enabled If checked, shift 1 will be included during scheduling. If not checked, shift 1 will be scheduled around. To inherit the shift enabled from the from the *area*, select the "Inherit From Parent" option.

Default Start Time The time of day that first shift starts. The first shift ends at the start of second shift. To inherit the time of day that first shift starts setting from the *area*, select the "Inherit From Parent" option.

Shift 2

Default Enabled If checked, shift 2 will be included during scheduling. If not checked, shift 2 will be scheduled around. To inherit the shift enabled from the from the *area*, select the "Inherit From Parent" option.

Default Start Time The time of day that second shift starts. The second shift ends at the start of third shift. To inherit the time of day that second shift starts setting from the *area*, select the "Inherit From Parent" option.

Shift 3

Default Enabled If checked, shift 3 will be included during scheduling. If not checked, shift 3 will be scheduled around. To inherit the shift enabled from the from the *area*, select the "Inherit From Parent" option.

Default Start Time The time of day that third shift starts. The third shift ends at the start of first shift. To inherit the time of day that third shift starts setting from the *area*, select the "Inherit From Parent" option.

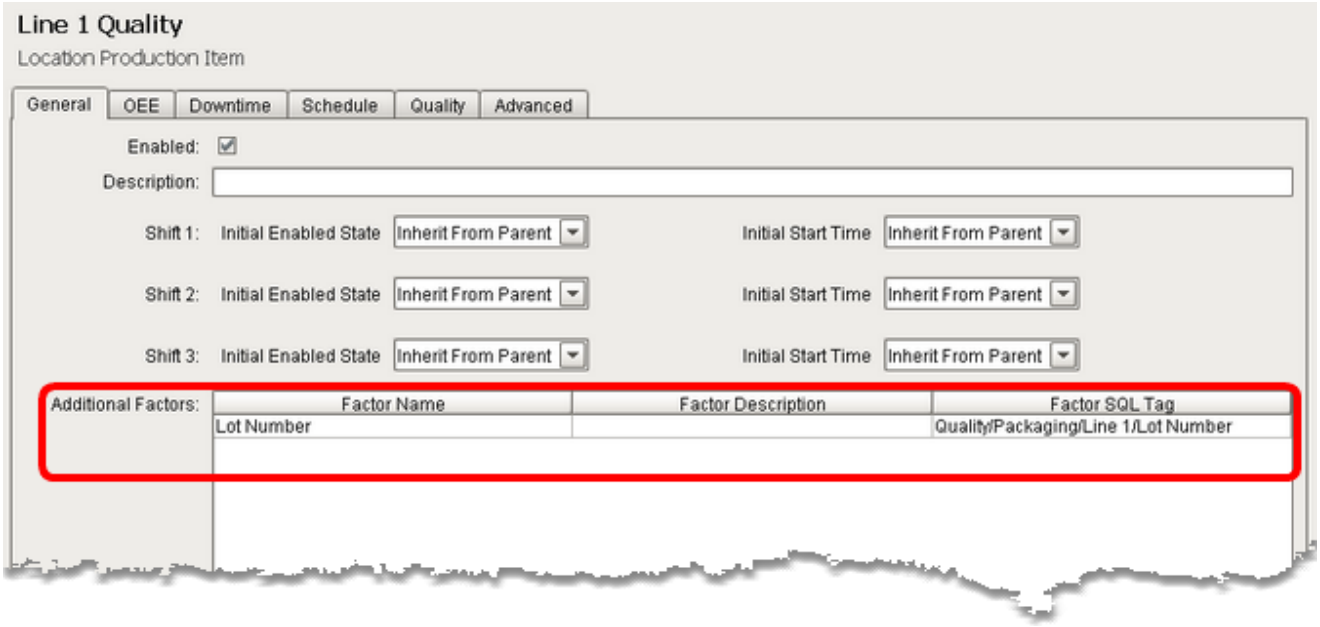
Additional Factors Additional Factors are user defined data points that are logged along with the production and downtime information. Once they are logged, they can be shown in charts, tables and reports. Additionally, other analysis can be done by filtering and/or setting up comparisons by their values.

Any value that can be read from an Ignition SQLTag can be added as a additional factor. This includes, values from barcode readers, databases, calculations, PLCs, or values derived from scripts, etc.

Example: An additional factor named cardboard manufacturer can be added. The operator can select the manufacturer that provided the cardboard or it can be obtained from some other source. Now, SPC results can be shown for each cardboard manufacturer. This can identify problems with raw material that directly affect quality.

Below is an example of an operator additional factor. The operators name will be logged along with the production and downtime data. By doing so, OEE and downtime information can be filtered and grouped by the operator name.

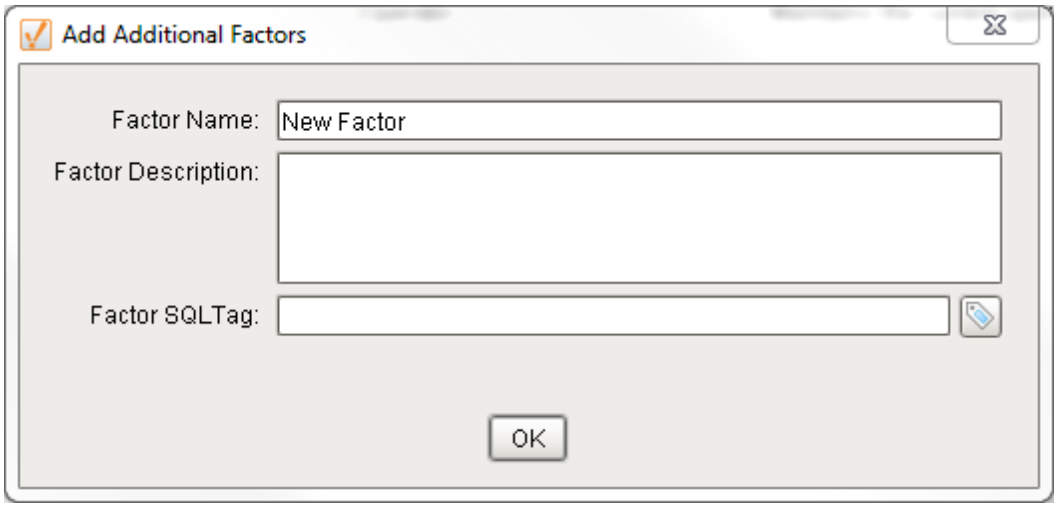
© Inductive Automation



Additional Factor List

Adding an Additional Factor

To add an additional factor, right-click anywhere on the additional factor table and select the **New** menu item. A dialog box will appear to allow entry of a new additional factor as shown below.



Additional Factor Settings

Factor Name

The required name of the additional factor is used to reference one additional factor from another. You can have any number of additional factors, but user usability will be hindered if too many are added. This is because the additional factors are added to user menus and if too many are added, the menus can become too long and confuse the end user.

The name given to an additional factor should be meaningful to the end user. Again, this is because additional factors appear in menus allowing the end user to filter and group analysis and report data by them.

Factor Description

© Inductive Automation

The optional description is just for reference or to keep internal notes about the additional factor.

Factor SQLTag

The required SQLTag is the source of the data value that will be logged. It is an Ignition SQLTag and the values can come from a PLC, a database query, other device in the field such as a barcode reader, expression, user input, or script. This opens the door to mesh any type of outside data into the MES module analysis and reporting.

Any type (format) of data that can be stored in an SQLTag can be logged. If SQLTag value is a string, then the end user can filter and group by the additional factor. If the SQLTag is a number, the option to filter and group by the additional factor will not be shown to the end user.

The SQLTag can be manually typed or pasted into the Factor SQLTag edit box. Optionally, clicking on the icon will display a browser where a SQLTag can be selected.

Editing an Additional Factor

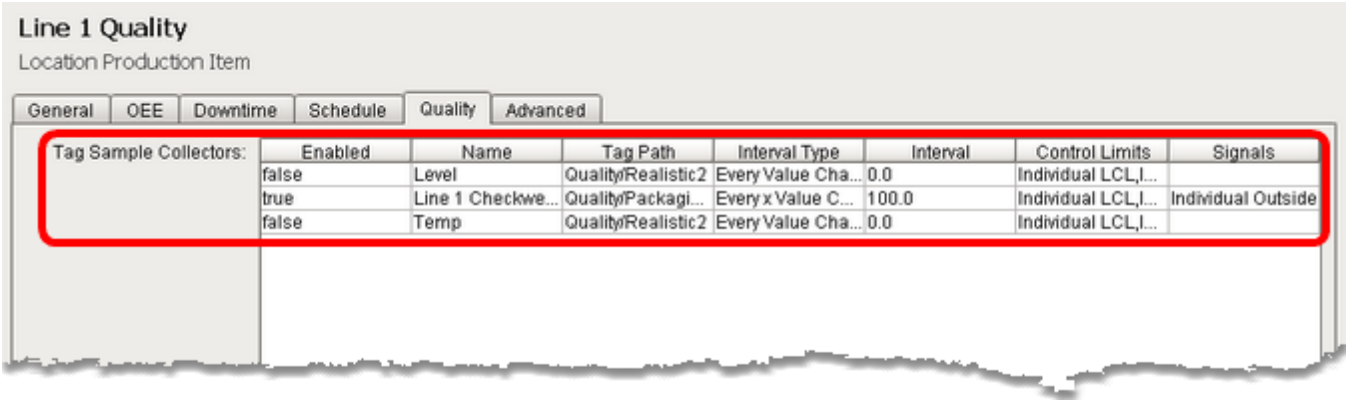
To edit an existing additional factor, right-click on the desired entry in the additional factor table and select the **Edit** menu item. A dialog box similar to the add dialog box will appear, allowing editing of the additional factor.

Deleting an Additional Factor

To remove an existing additional factor entry, right-click on the desired entry in the additional factor table and select the **Delete** menu item. A window will appear confirming that you want to remove the additional factor. The additional factor will no longer be logged. However, any production runs that occurred before the additional factor was deleted, will still show in the analysis and reporting.

Location Quality Settings

These settings can be found in the Location folder under the "Quality" tab. The SQLTag Sample Collectors allow for the automatic collection of sample data. For more information, see Tag Sample Collectors.



Location Quality Settings Screen

See SQLTag Sample Collectors for more information.

3.3.3 Control Limits

3.3.3.1 Overview

Control limits are upper (UCL) and lower (LCL) values that are calculated from the data that is gathered from a process. These limits, typically shown as horizontal lines on the control charts, reflect the past performance of that process. For the p and u Charts, the control limits can vary for each sample depending on the number of items inspected for each sample. See the SPC Charts in the Introduction sections for more information. In the SPC Quality Module, these limits can be calculated automatically, or entered manually. In the SPC module, control limits can be either calculated or can act as specification limits. Specification limits are requirements made by the company, not a reflection of the process itself.

There are different control limits types for each type of control chart. For example, the XBar only supports XBar UCL, XBar LCL and XBar Other control limits types and cannot be calculated or shown for any other control chart besides the XBar control chart.

The control limits are defined by the enterprise and can be added, edited or deleted on the Enterprise page in the designer under the "Quality" tab. By default, the standard control limits are added when a new Enterprise Production Item is added.

3.3.3.2 Default Control Limits

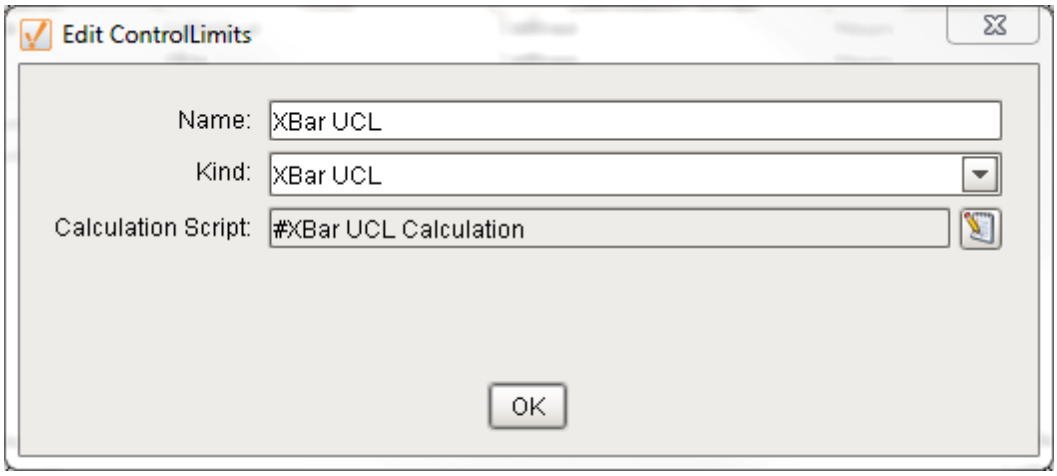
When a new Enterprise Production Item is added, the following control limits are added:

c LCL
c UCL
Histogram LCL
Histogram UCL
Individual LCL
Individual UCL
Median LCL
Median UCL
MR LCL
MR UCL
np LCL
np UCL
p LCL
p UCL
Range LCL
Range UCL
StdDev LCL
StdDev UCL
StdDev XBar LCL
StdDev XBar UCL
u LCL
uUCL
XBar LCL
XBar LSL
XBar UCL
XBar USL

3.3.3.3 Add Control Limits

To add a control limit, right-click the Control Limits table and select New from the drop-down menu. A window will appear with several fields to be completed, including the name and kind of the control limit, as well as the scripting necessary to use the control limit.

© Inductive Automation



Adding a Control Limit

Name

This is the required unique name of the control limit as it will appear in selection lists and control charts. It is better to keep this short in length so that it will fit better on the control charts.

Kind

Each type of control chart has control limit kinds that it works with. If a control limit will be used with a Individual control chart, then either the Individual LCL (lower control limit), Individual UCL(upper control limit) or Individual Other control limit kinds must be used.

Available control limits kinds grouped by control chart type:

XBar	XBar, Xbar S
XBar UCL	
XBar LCL	
XBar Other	
XBar Range	XBar
Range LCL	
Range UCL	
Range Other	
Individual	Individual
Individual LCL	
Individual UCL	
Individual Other	
Moving Range	Individual, Median
MR LCL	
MR UCL	
MR Other	
Standard Deviation	XBar S
Standard Deviation LCL	
Standard Deviation UCL	
Standard Deviation Other	
Median	Median
Median LCL	
Median UCL	

© Inductive Automation

Median Other

p p Chart
p LCL
p UCL
p Other

np np Chart
np LCL
np UCL
np Other

u u Chart
u LCL
u UCL
u Other

c c Chart
c LCL
c UCL
c Other

Histogram Histogram
Histogram LCL
Histogram UCL
Histogram Other

Calculation Script

Because control limit calculations can vary, the SPC module uses scripting. This allows the user to override the default calculation of a control limit or add new control limits that the SPC module may not provide by default. Additionally, they can be removed, cleaning up selection lists of control limits that may never be used.

When a user or script function is used to initiate a control limit to be calculated, the script in the associated control limit is executed. An event object is passed into the script that contains the information and data to calculate the new control limit value. We will introduce this event here, but see Control Limit Event object for more information.

In the example below, any lines that start with the pound (#) character are comments and are ignored when the script is executed.

The event.getData() on line 8, returns the samples that will be used to calculate the new control limit. It is a data set (see Ignition DataSet in scripting for more information) and contains a row of data for each sample. Each sample row includes measurement values, calculated values (such as xBar, standard deviation, etc), sample date and time. For the p and u charts where the control limits can vary by sample, this data set includes columns to which the the newly calculated control limit for each sample can be saved.

The ds.getColumnIndex on lines 11 and 12, returns the column number of the "XBar" and "Range" columns. This is done for speed reasons because it is faster to reference the column by number instead of finding the column by name.

From line 19 to 21, each sample row in the data set is cycled through. This is done to total the xBar and range values. The ds.getValueAt() function returns the value in the data set for the specified row and column.

© Inductive Automation

Line 24 calculates the average of the xBar values, also known as x double bar (XDBar).
Line 25 calculates the average of the range values, also known as range bar (RBar).

The event.getSampleSize() in line 28, returns the number of measurements per sample. This will be used to determine which a2 value to use from the array in line 5. The a2 is a factor to calculate the 3 sigma or 3 times standard deviation value and changes based on the number of measurements in each sample.

Lines 31 through 34 lookup the a2 value that is going to used to calculate the new control limit value. A quick range check is done to prevent reading a value that is outside of the array limits.

Line 37 calculates the new UCL value.

And finally, the value is saved to pass back the new control limit value in line 40.

Default XBar UCL control limit calculation script:

```
1  #XBar UCL Calculation
2  #Define the A2 factors array.
3  #The A2 factors correspond to the sample size which starts at 2.
4  #This is why element 0 and 1 of the array are 0.
5  a2 = [0.0, 0.0, 1.880, 1.023, 0.729, 0.577, 0.483, 0.419, 0.373, 0.337, 0.308, 0.285, 0.266, 0.249, 0
6
7  #Get the SPC data that the XBar UCL will be calculated for
8  ds = event.getData()
9
10 #Get the columnn indexes within the SPC data
11 xBarColNdx = ds.getColumnIndex("XBar")
12 rangeColNdx = ds.getColumnIndex("Range")
13
14 #Initialize xBar and range sums that are need to calculate average xBar and range.
15 xBarSum = 0.0
16 rSum = 0.0
17
18 #Cycle through each row and add to the sums
19 for row in range(ds.rowCount):
20        xBarSum = xBarSum + ds.getValueAt(row, xBarColNdx)
21        rSum = rSum + ds.getValueAt(row, rangeColNdx)
22
23 #Calculate the average xBar and range
24 xDBar = xBarSum / ds.rowCount
25 rBar = rSum / ds.rowCount
26
27 #Get the sample size.
28 sampleSize = event.getSampleSize()
29
30 #Lookup the A2 value
31 if sampleSize < len(a2):
32        a2Value = a2[sampleSize]
33 else:
34        a2Value = a2[len(a2) - 1]
35
36 #Calculate the xBar UCL
37 ucl = xDBar + a2Value * rBar
38
39 #Return the new xBar UCL back to the SPC module
40 event.setControlLimitValue(ucl)
```

Looking at the default control limit calculations along with the Scripting section of this manual and the

Scripting section in the Ignition manual is the best method to learn all the possibilities of calculating control limits.

3.3.3.4 Edit Control Limits

To edit a control limit, right-click the Control Limits table and select Edit from the drop-down menu. A window will appear identical to the window used to add control limits. Once the desired fields have been edited, select OK.

For more information see Add Control Limit section.

3.3.3.5 Delete Control Limits

To delete a control limit, select the item to be deleted. After selecting, right-click the item and select Delete from the drop-down menu. A window will appear confirming that you permanently want to delete the control limit.

3.3.3.6 Import/Export

To export control limit entries, right-click anywhere on the table containing control limit entries and select the **Export** menu item. A dialog box will appear to allow selection of an existing file or the entry of a name for the new file to which the control limit entries are saved. If a file extension is not entered, then the default .csv will be used.

The first line of the file must contain at least the property names separated by commas. If additional names exist, they will be ignored. The property names can be in any order. Below is a sample csv file showing multiple control limit entries. The lines in the example shown below have been shortened.

© Inductive Automation

```
Name,Kind,Script
"c LCL","23","#c LCL Calculation\nimport math\n\n#Get the
"c UCL","22","#c UCL Calculation\nimport math\n\n#Get the
"Histogram LCL","29",""
"Histogram UCL","28",""
"Individual LCL","11","#Individual LCL Calculation\n#Get t
"Individual UCL","10","#Individual UCL Calculation\n#Get t
"Median LCL","14","#XBar LCL Calculation\n#Define the A2 f
"Median UCL","13","#XBar UCL Calculation\n#Define the A2 f
"MR LCL","35","#Moving Range LCL Calculation\n#The LCL for
"MR UCL","34","#Moving Range UCL Calculation\n#Get the SPC
"np LCL","20","#np LCL Calculation\nimport system\nimport
"np UCL","19","#np UCL Calculation\nimport math\n\n#Get th
"p LCL","17","#p LCL Calculation\nimport system\nimport ma
"p UCL","16","#p UCL Calculation\nimport math\n\n#Get the
"Range LCL","5","#Range UCL Calculation\n#Define the D3 fa
"Range UCL","4","#Range UCL Calculation\n#Define the D4 fa
"StdDev LCL","8","#Standard Deviation LCL Calculation\n#De
"StdDev UCL","7","#Standard Deviation UCL Calculation\n#De
"StdDev XBar LCL","2","#Standard Deviation XBar LCL Calcul
"StdDev XBar UCL","1","#Standard Deviation XBar UCL Calcul
"u LCL","26","#u LCL Calculation\nimport math\n\n#Get the
"u UCL","25","#u UCL Calculation\nimport math\n\n#Get the
"XBar LCL","2","#XBar LCL Calculation\n#Define the A2 fact
"XBar LSL","3",""
"XBar UCL","1","#XBar UCL Calculation\n#Define the A2 fact
"XBar USL","3",""
```

To import downtime entries, right-click anywhere on the control limit table and select the **Import** menu item. A dialog box will appear to allow selection of a comma separated values (csv) formatted file.

3.3.4 Out of Control Signals

3.3.4.1 Overview

Out-of-control signals occur in a variety of situations, but all the signals indicate a change in the process where it is considered to be abnormal, or out of control. Some signals include: six points in a row that are increasing or decreasing, eight points in a row that are farther than one standard deviation away from the centerline, or fourteen points in a row that are alternating up and down. When used properly, these signals can identify important changes that can help to improve or maintain the process.

Signals can be configured so that they are evaluated every time new sample data is recorded. This allows for quick and automatic detection of out of control conditions. Once an out of control condition is automatically detected, Ignition provides a variety of actions that can be performed, such as standard alerting, communications, logging and more.

© Inductive Automation

For automatic signal evaluation to be enabled, the Look Back Period must be set to something other than "No Auto Evaluation", a valid look back duration must be set and the signal must be selected for the desired sample definitions.

Out of Control Signals can be added, edited or deleted on the Enterprise page in the designer under the "Quality" tab.

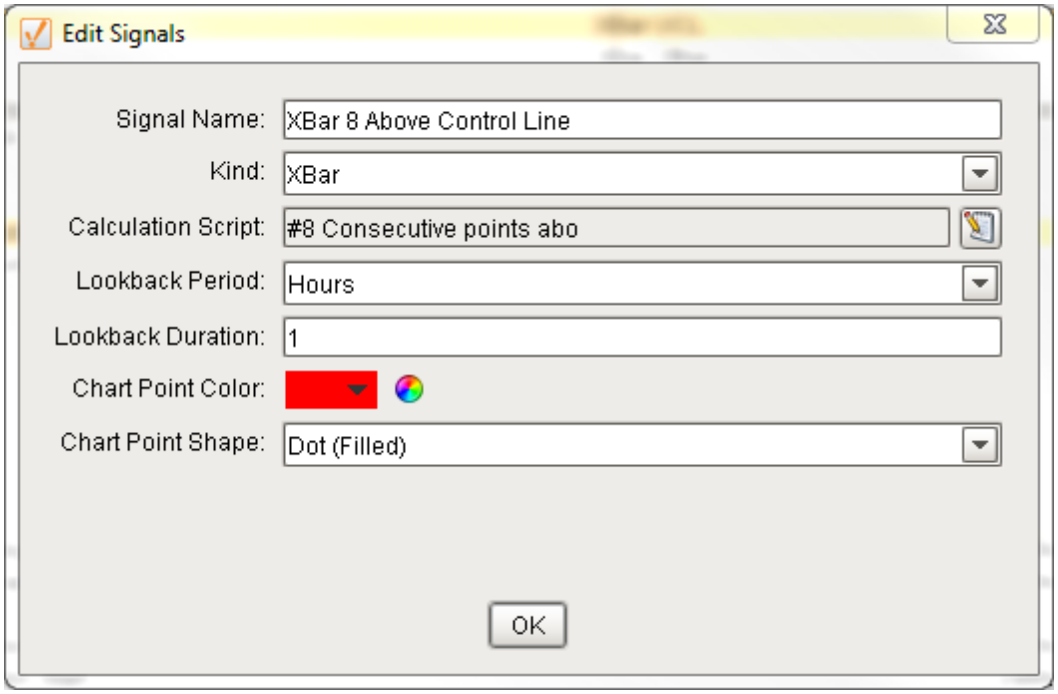
3.3.4.2 Default Signals

When a new Enterprise Production Item is added, the following control limits are added:

Individual Outside
Out of Limits
Outside Limits

3.3.4.3 Add Signals

To add an out of control signal, right-click the Out of Control Signals table and select New from the drop-down menu. A window will appear with several fields to be completed, including the signal name, kind, calculation script, lookback period, lookback duration, chart point color and chart point shape.



Adding a Signal

Signal Name

This is the required unique name of the signal as it will appear in selection lists and control charts. It is better to keep this short in length so that it will fit better on the control charts.

Kind

Each type of control chart has signal kinds that it works with. If a signal will be used with a Individual control chart, then the Individual signal kind must be used.

Available control limits kinds grouped by control chart type:
XBar
Range

© Inductive Automation

Individual
MR
Standard Deviation
Median
p
np
u
c

Calculation Script

Because signal calculations can vary, the SPC module uses scripting. This allows the user to override the default calculation of a signal or adding new signals that the SPC module may not provide by default. Additionally, they can be removed, cleaning up selection lists of signals that may never be used.

Signals are evaluated when viewing them on control charts or when new sample data is recorded. When either of these trigger the signals to be calculated, the script in the associated signal is executed. An event object is passed into the script that contains the information and data to calculate the signal state values. We will introduce this event here but see Signal Event object for more information.

In the example below, any lines that start with the pound (#) character are comments and are ignored when the script is executed.

Line 2 initializes a variable used to track how many consecutive calculated values (like the x bar value) are above the control line (like the x double bar value).

The event.getData() on line 5, returns the samples that will be used to calculate the signal state values. It is a data set (see Ignition DataSet in scripting for more information) and contains a row of data for each sample. Each sample row includes measurement values, calculated values (such as xBar, standard deviation, etc), sample date and time and control limits. There is also a column named the sample as the signal to save the signal state value. By setting the value of this column to a zero (0), the sample is in control for this signal, and by setting the value of this column to a one (1), the sample is out of control.

The ds.getColumnIndex on lines 8 through 10, returns the column number of the "XBar", "XDBar" and signal result columns. This is done for speed reasons because it is faster to reference the column by number instead of finding the column by name.

Starting with line 13, each sample row in the data set is cycled through.

Line 16 reads the calculated value that in this case is the xBar value.
Line 17 reads the average of the calculated values, which in this case is the xDBar value.

In line 20, a test is done for the xBar value being greater than the xDBar. If it is, further checking is done in lines 22 through 38. If it is not, then the consecutive count variable is reset and the signal state value is set to 0 for the sample in lines 42 and 43.

Line 22 adds to the consecutive count variable before checking if the threshold of 8 has been exceeded.

Line 25 checks if the consecutive count threshold has been exceed. If not, the signal state value for the sample is set to 0 and the consecutive count variable is left at its current value.

Line 28 checks if the consecutive count just exceeded the threshold. If it just did, the signal state values for the previous 8 samples are set to 1. This flags the current sample and the previous 7 samples as out

© Inductive Automation

of control.

The else statement in line 35 is a check that occurs if more than 8 consecutive xBar values exceed the xBar value. It sets the signal state value to 1 and leaves the consecutive count variable at its current value.

Default 8 consecutive points above control limit signal calculation script:

```
1  #8 Consecutive points above control line signal calculation
2  consecutiveCount = 0
3
4  #Get the SPC data that the signal will be calculated for
5  ds = event.getData()
6
7  #Get the columnn indexes within the SPC data
8  xBarColNdx = ds.getColumnIndex("XBar")
9  xDBarColNdx = ds.getColumnIndex("XDBar")
10 resultColNdx = ds.getColumnIndex("XBar 8 Above Control Line")
11
12 #Cycle through each row and check signal
13 for row in range(ds.rowCount):
14
15        #Get the values to compare
16        xBar = ds.getValueAt(row, xBarColNdx)
17        xDBar = ds.getValueAt(row, xDBarColNdx)
18
19        #Test if the x bar value is above x double bar value
20        if xBar > xDBar:
21               #Add to the consecutive count
22               consecutiveCount = consecutiveCount + 1
23
24               #Test if less than 8 consecutive x bar values are above x double bar
25               if consecutiveCount < 8:
26                      #Write a zero to the result column, meaning we are in control
27                      ds.setValueAt(row, resultColNdx, 0)
28               elif consecutiveCount == 8:
29                      #Now 8 consecutive x bar values are above the x double bar
30                      #Write a 1 into the last 8 row because, they are all out of control
31                      ndx = row
32                      while ndx > 0 and ndx > row - 8:
33                             ds.setValueAt(ndx, resultColNdx, 1)
34                             ndx = ndx - 1
35               else:
36                      #Over 8 consecutive x bar values are above x double bar
37                      #Continue writing a 1 into the result because this row is still out of control
38                      ds.setValueAt(row, resultColNdx, 1)
39        else:
40               #x bar value is below, reset the consecutive count
41               #and write a zero to the result column, meaning we are in control
42               consecutiveCount = 0
43               ds.setValueAt(row, resultColNdx, 0)
```

Look Back Period

This property defines the time units of the Look Back Duration property.

No Auto Evaluation	Disable automatic signal evaluation after new sample data is recorded.
Seconds	
Minutes	
Hours	

Days
Months

Look Back Duration
When automatic signal evaluation is used, this property, along with the Look Back Period property, defines the time range of samples to pass to the calculation script. The calculation script can then cycle through the range of samples to find out of control conditions.

Chart Point Color
For samples that are out of control, this is the color to display the sample value on the control charts.

Chart Point Shape
For samples that are out of control, this is the shape to display for sample value on the control charts.

Looking at the default signal calculations along with the Scripting section of this manual and the Scripting section in the Ignition manual is the best method to learn all the possibilities of calculating signals.

View the section on Scripting for more information.

3.3.4.4 Edit Signals

To edit an out of control signal, right-click the Out of Control Limits table and select Edit from the drop-down menu. A window will appear identical to the window used to add out of control limits. Once the desired fields have been edited, select OK.

For more information see Add Signals section.

3.3.4.5 Delete Signals

To delete an out of control signal, select the item to be deleted. After selecting, right-click the item and select Delete from the drop-down menu. A window will appear confirming that you permanently want to delete the out of control signal.

3.3.4.6 Import/Export

To export signal entries, right-click anywhere on the table containing signal entries and select the **Export** menu item. A dialog box will appear to allow for the selection of an existing file or the entry of a name for the new file to which the out of control signal entries are saved. If a file extension is not entered, then the default .csv will be used.

The first line of the file must at least contain the property names separated by commas. If additional names exist, they will be ignored. The property names can be in any order. Below is a sample csv file showing multiple signal entries. The lines in the example shown below have been shortened.

```
SignalName,SignalKind,SignalScript,SignalAutoEvaluatePeriod,SignalAutoEvaluateDuration,SignalChartColor,S
"Individual Outside","5","ds = event.getData()\nxBarColNdx = ds.getColumnIndex(\"XBar\")\nuclColNdx = ds.
"Out of Limits","1","ds = event.getData()\nrangeColNdx = ds.getColumnIndex(\"XBar\")\nuclColNdx = ds.getC
"Outside Limits","2","ds = event.getData()\nrangeColNdx = ds.getColumnIndex(\"Range\")\nuclColNdx = ds.ge
"XBar 8 Above Control Line","1","#8 Consecutive points above control line signal calculation\nconsecutive
"XBar 8 Below Control Line","1","#8 Consecutive points below control line signal calculation\nconsecutive
```

3.3.5 Sample Intervals

3.3.5.1 Overview

Samples can always be taken manually, but the SPC module supports scheduling samples to be taken manually and automatically taking samples.
Sample Intervals are used to define the amount of time or number of readings that pass between samples. For example, the interval may be a timed interval that occurs every three minutes, every 100 readings, or samples can be taken continuously. These options will be available when defining a sample on the Sample Definition page when adding or editing a location. They also are used by Tag Sample Collectors.

Sample Intervals can be added, edited or deleted on the Enterprise page of the designer under the "Quality" tab.

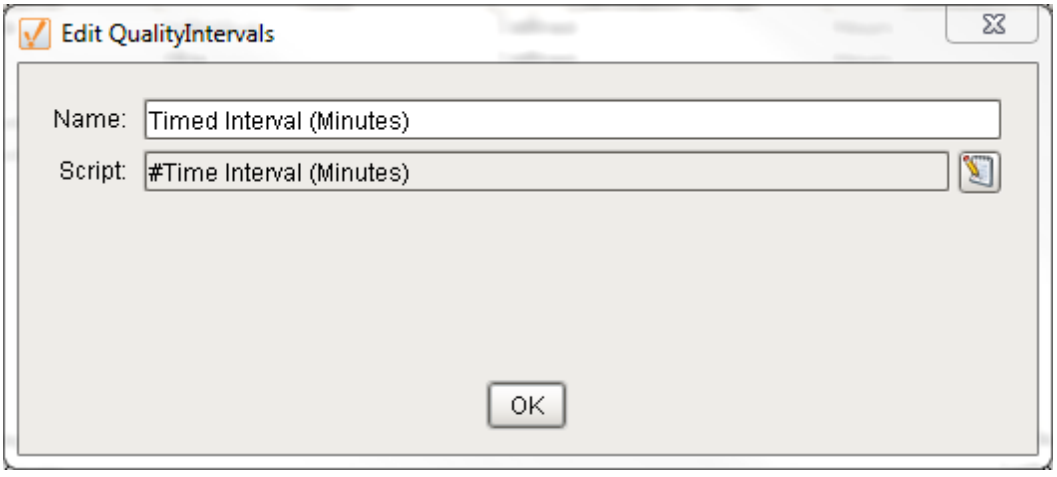
3.3.5.2 Default Intervals

When a new Enterprise Production Item is added, the following intervals are added:

Every Value Change
Every x Value Change
Manual
Once at Production End
Once at Production Start
Shift Change
Timed Interval (Days)
Timed Interval (Hours)
Timed Interval (Minutes)
Timed Interval (Seconds)

3.3.5.3 Add Intervals

To add a sample interval, right-click the Sample Intervals table and select New from the drop-down menu. A window will appear with several fields to be completed, including the name of the sample interval, as well as the scripting necessary to use the sample interval.



Name
This is the required unique name of the interval as it will appear in selection lists.

Script

Because the default intervals may not be exactly what you are looking for, the SPC module uses scripting. This allows the user to override the default calculation of an interval or adding new intervals that the SPC module may not provide by default. Additionally, they can be removed, cleaning up selection lists of intervals that may never be used.

In the sample definition, an interval can be selected and will define when new samples are scheduled. These scheduled samples require manual entry of measurements.

In the Tag Sample Collector configuration, an interval is used to define when to automatically add new samples.

In the example below, any lines that start with the pound (#) character are comments and are ignored when the script is executed.

Line 2 will allow us to use the Calendar object to do math with date values. See the Ignition documentation for more information.

Line 5 returns the seconds since the last time a sample was scheduled. There is a wealth of information in the event object that can be used to determine if a sample should be scheduled or taken. See Interval

Line 8 returns the duration to use. In this case it is in minutes

Line 9 returns the coming due minutes. It is going to be used to schedule a sample prior to the time it is due, so that it will show in the sample list component prior to the time it is actually due. For automatic Tag Sample Collectors, the coming due will be 0 and the sample will be recorded and measurements collected when the sample is created.

Line 12 does the actual checks to determine if a new sample should be scheduled. If `secSinceLastSample` equals `None`, then it means a sample has not been scheduled for the sample definition and location that is being checked. In this case, a new sample should be created.

Lines 15 through 17 calculate the scheduled start time for the sample. This is the time that the sample will appear in the sample list component and set the Sample Coming Due tag associated with the production location.

Line 20 sets the create sample flag that tells the SPC module to create a new sample after executing this script. This can be done through script functions specifically for creating samples, but this simplifies the task of doing so down to one line of script.

Time Interval (Minutes) script:

```
1   #Time Interval (Minutes)
2   from java.util import Calendar
3
4   #Get the last time a sample was scheduled
5   secSinceLastSample = event.getSecSinceLastSampleScheduled()
6
7   #Calculate the interval in seconds
8   intervalSec = event.getInterval() * 60
9   comingDueSeconds = event.getComingDueMin() * 60
10
11  #If a sample has not been scheduled or intervalSec has expired, schedule a new sample
12  if secSinceLastSample == None or secSinceLastSample >= intervalSec - comingDueSeconds:
13
```

© Inductive Automation

```
14        #Schedule next sample to start now + coming due minutes
15        cal = Calendar.getInstance()
16        cal.add(Calendar.SECOND, int(comingDueSeconds))
17        event.setScheduleStart(cal.getTime())
18
19        #Create new sample - no values are recorded
20        event.setCreateSample(1)
```

3.3.5.4 Edit Intervals

To edit a sample interval, right-click the Sample Intervals table and select Edit from the drop-down menu. A window will appear identical to the window used to add sample intervals. Once the desired fields have been edited, select OK.

3.3.5.5 Delete Intervals

To delete a sample interval, select the item to be deleted. After selecting, right-click the item and select Delete from the drop-down menu. A window will appear confirming that you permanently want to delete the sample interval.

3.3.5.6 Import/Export

To export interval entries, right-click anywhere on the table containing interval entries and select the **Export** menu item. A dialog box will appear to allow selection of an existing file or the entry of a name for the new file to which the interval entries are saved. If a file extension is not entered, then the default .csv will be used.

The first line of the file must at least contain the property names separated by commas. If additional names exist, they will be ignored. The property names can be in any order. Below is a sample csv file showing multiple interval entries. The lines in the example shown below have been shortened.

```
QualityIntervalName,QualityIntervalScript
"Every Value Change","#Record sample every time a tag value changes\nif event.isVal
"Every x Value Changes","#Every x value changes\nif event.isValueChangedEvent():\n\
"Manual",""
"Once at Production End","#Once at production enb \nif event.isTraceEndedEvent() ==
"Once at Production Start","#Once at production start \nif event.isTraceStartedEver
"Shift Change","#Once at shift change \nif event.isShiftChangeEvent() == 1:\n\tever
"Timed Interval (Days)","#Time Interval (Days)\nfrom java.util import Calendar\n\nf
"Timed Interval (Hours)","#Time Interval (Hours)\nfrom java.util import Calendar\n\
"Timed Interval (Minutes)","#Time Interval (Minutes)\nfrom java.util import Calenda
"Timed Interval (Seconds)","#Timed Interval (Seconds)\n#Test if product is being ru
```

3.3.6 SQLTag Sample Collectors

3.3.6.1 Overview

Tag Sample Collectors are used to automatically collect measurement data from an Ignition tag and create samples with the collected measurement data. When configuring, the selected interval defines how often to create a new sample. For example, on every 100th value change of a checkweigher value, create a new sample and record the current value. Or, every 10 minutes while a process is running, create a sample and record the current temperature.

The measurement data can come from a variety of sources including any OPC connected device, values from external databases, manual entries, etc.

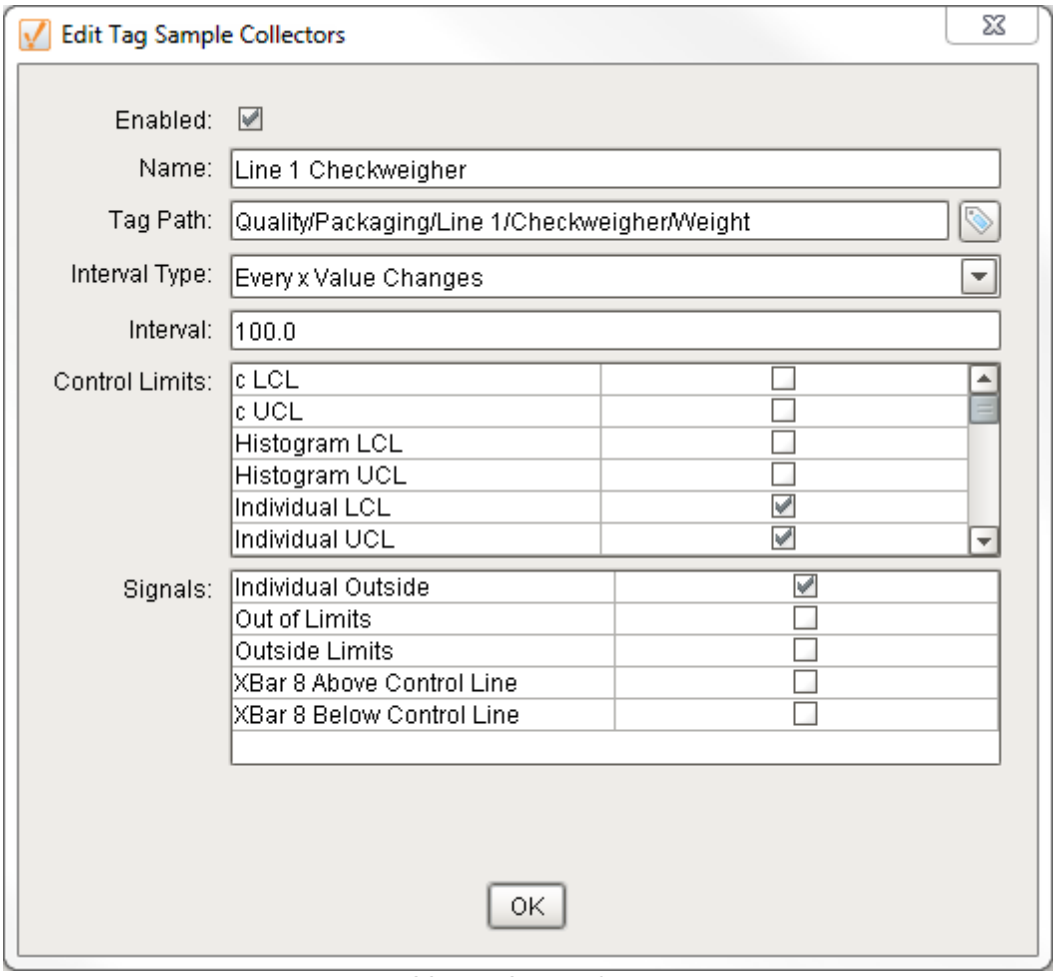
© Inductive Automation

Any samples that are automatically created and recorded by a Tag Sample Collector are automatically approved and will appear in the control charts. By setting the Auto Refresh property of either the SPC Selector or SPC Controller components, new samples will appear in the control charts in real time as they are created. In addition, the appropriate events found on the Advanced tab for the production location will be executed.

Tag Sample Collectors can be added, edited or deleted on the Location page of the designer under the "Quality" tab.

3.3.6.2 Add Sample Collectors

To add a Tag Sample Collector, right-click the Tag Sample Collector table and select New from the drop-down menu. A window will appear with several fields to be completed, including the name of the tag sample collector, as well as the tag path and other properties required.



Add SQL Tag Sample Collector

Enabled

Tag Sample Collectors enabled property provides a method of stopping the automatic collection of measurements and creation of samples. Additionally, any tags associated with this property can be changed to start and stop automatic collection. See Quality OPC Values for more information.

Name
This is the required unique name of the Tag Sample Collector as it will appear, with "SQLTag-" pre pended to it, in selection lists. Behind the scenes, a sample definition is created using this sample name. Sample definitions created for the purpose of Tag Sample Collectors will not appear in the definition management and manual sample entry client screens.

SQLTag Path
This is the SQLTag path from which measurement values will be read.

Interval Type
The interval options that can be selected here match those defined in the Intervals list on the Enterprise quality tab. Only intervals that have script will be included as options for Tag Sample Collectors. The reason for this is that manual intervals, which are the those without script, will never be created and do not apply to automatic collection of measurements.

Interval
The interval to collect data and create new samples. The units of this interval are defined by the interval type and can be minutes, days, every x value read, etc.

Control Limits
The control limits that are checked will be calculated for this Tag Sample Collector during signal evaluations. Available control limit options are defined in the Control Limits list on the Enterprise quality tab. It is important to include control limits that a signal depends on or the signal will not be evaluated correctly.

Signals
The signals that are checked will be evaluated every time a new sample is recorded by the Tag Sample Collector. Available signal options are defined in the Signals list on the Enterprise quality tab.

3.3.6.3 Edit Sample Collectors

To edit a tag sample collectors, right-click the Tag Sample Collector table and select Edit from the drop-down menu. A window will appear identical to the window used to add tag sample collector. Once the desired fields have been edited, select OK.

3.3.6.4 Delete Sample Collectors

To delete a tag sample collector, select the item to be deleted. After selecting, right-click the item and select Delete from the drop-down menu. A window will appear confirming that you permanently want to delete the tag sample collector.

3.3.6.5 Import/Export

To export tag sample collector entries, right-click anywhere on the table containing tag sample collector entries and select the **Export** menu item. A dialog box will appear to allow selection of an existing file or the entry of a name for the new file to which the collector entries are saved.. If a file extension is not entered, then the default .csv will be used.

The first line of the file must at least contain the property names separated by commas. If additional names exist, they will be ignored. The property names can be in any order. Below is a sample csv file showing multiple tag sample collector entries. The lines in the example shown below have been shortened.

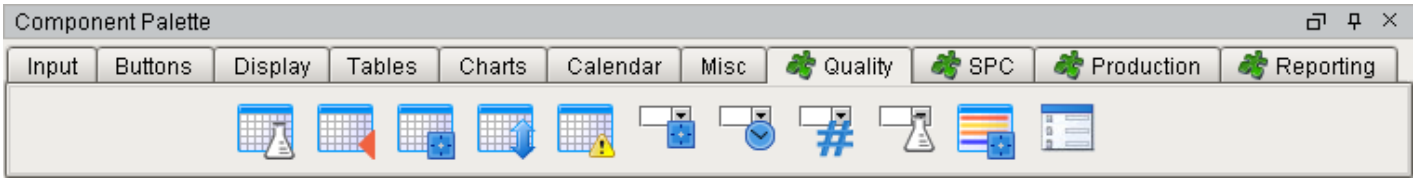
```
Enabled,Name,SQLTag Path,Interval Type,Interval,Control Limits,Signals
"true","Line 2 Checkweigher","Quality/Packaging/Line 2/Checkweigher/Weight","Every x Valu
```

© Inductive Automation

3.4 Component Reference

This section is a reference for all of the components that come with the SPC Module.

3.4.1 Quality Components



Quality Components

3.4.1.1 Definition List

Description

A component that provides a list of sample definitions. A sample definition defines the attributes (measurements), locations, control limits and out of control signals to use for samples. It allows for adding, editing and deleting samples and works with the Definition Attribute List, Definition Location List, Definition Control Limit List and Definition Signals List components.

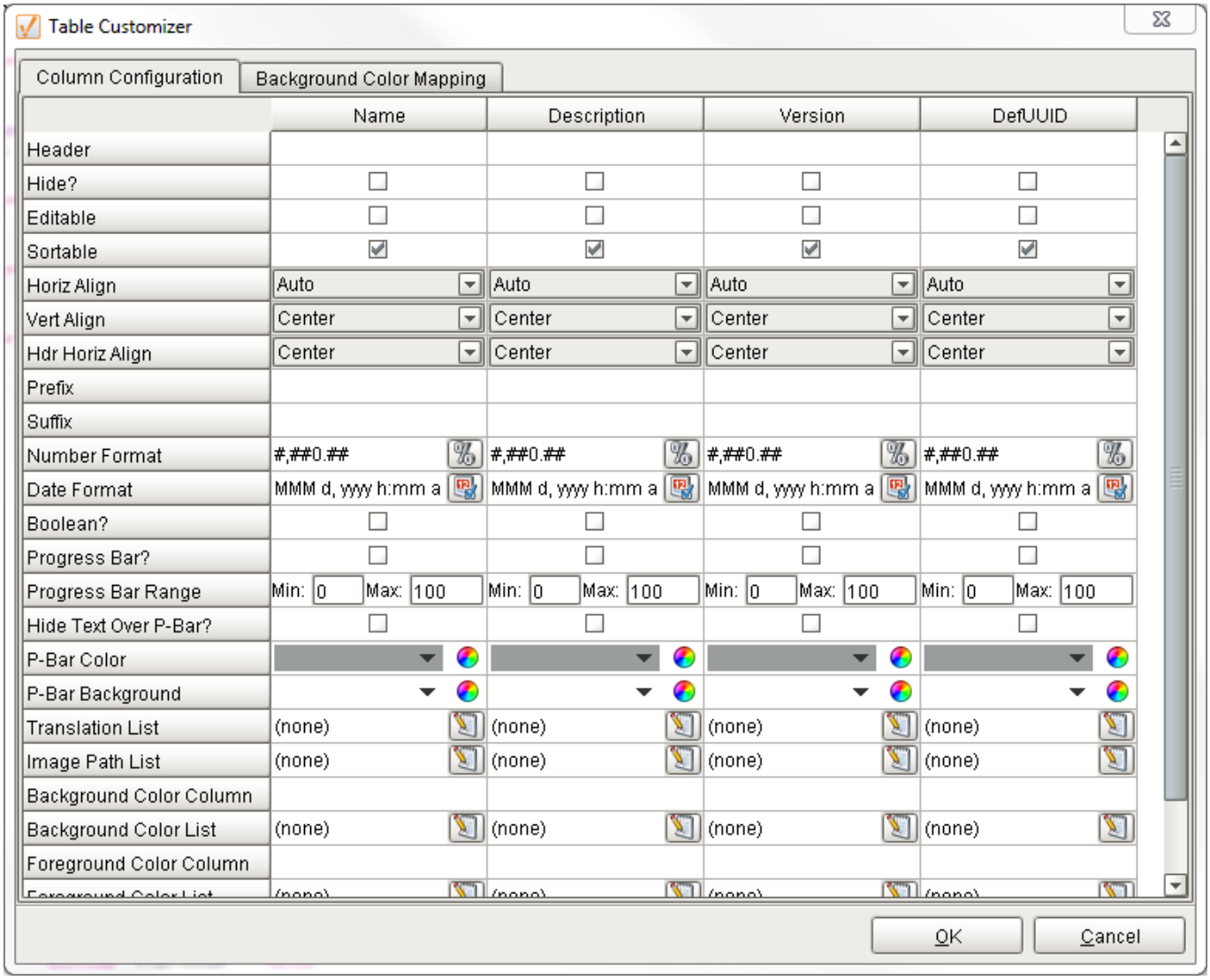
There is no need for SQL queries or scripting to display sample definitions. The SPC Module will send notifications to each client with a Definition List component being displayed when there is a change to any sample definitions made by another user. This event-based functionality optimizes updates, reducing database updates and network bandwidth.

Name	Version
Final	1
Final Imperfections	1
pH	1
pH2	1
Product Moisture	1
Value Inspection	1
Viscosity	1

Sample Definition List

The Ignition table customizer is used to change the appearance of the table. To access the customizer, right-click on the Definition List component and select the Cutomizers->Table Customizer menu item. Using the customizer, you can hide columns, change colors, and change formatting to make the Definition List appear as desired.

© Inductive Automation



Ignition Table Customizer

Properties

This component has standard Ignition properties with the addition of the following properties:

Show Disabled When set to true, disabled sample definitions will be shown. This provides a method to re-enable previously disabled sample definitions.

```
Scripting name     showDisabled
Data Type          boolean
```

Read Only When set to true, prevents the popup menu from appearing when the user right-clicks on the Definition List component.

```
Scripting name     readOnly
Data Type          boolean
```

Activity Number of seconds to wait after user activity before automatic refresh of data.

```
There is no scripting support for this property.
```

Events

This component has standard Ignition events with the addition of the following events:

add Is fired when "Add" menu item is selected. The "Add" menu item will only appear if script has been added to this event.

Event Properties (none)

edit

Event Properties

event.getSampleDefinitionName() Return the currently selected sample definition name. String

remove

Event Properties

event.getSampleDefinitionName() Return the currently selected sample definition name. Data Type String

event.setRemoveDefinition (boolean) Used to tell the Definition List component to remove the selected sample definition. If this is not included in the remove event script with a parameter of 1, then the sample definition will have to be removed using another method.

event.setSuppressConfirmation By including this in the remove event script with a parameter of 1,

(boolean) the confirmation message will not be shown before removing a sample definition. Including the event.RemoveDefinition (0) and setSuppressConfirmation(1) script lines in the remove event will prevent default handling of sample definitions. This allows for custom handling of the removal of sample definitions.

Methods

`save()`

Save changes to the currently selected sample definition.

parameters (none)
returns nothing

`cancel()`

Undo the changes to the currently selected sample definition.

parameters (none)
returns nothing

`getSampleDefinition()`

Return the currently selected sample definition.

returns

Sample Definition — An instance of the currently selected sample definition
`Data Type` SampleDefinition
See SampleDefinition Object for more information.

`addSampleDefinition(sampleDefinition)`

Add the sample definition specified in the parameter.

parameters

`sampleDefinition` — Instance of the sample definition to add.
`Data Type` SampleDefinition
See SampleDefinition Object for more information.

returns

message — Contains a description of any error encountered, otherwise it will be empty
`Data Type` String

`updateSampleDefinition(sampleDefinition)`

Update the sample definition specified in the parameter.

parameters

`sampleDefinition` — Instance of the sample definition to update.
`Data Type` SampleDefinition
See SampleDefinition Object for more information.

returns

message — Contains a description of any error encountered, otherwise it will be empty

© Inductive Automation

Data Type String

refresh()

Refresh the currently selected sample definition. This causes any associated components such as the Definition Attribute List to also be refreshed.

parameters (none)
returns nothing

3.4.1.2 Definition Attribute List

Description

A component that provides a list of measurement attributes associated with a sample definition.

There is no need for SQL queries or scripting to display sample definition attributes. If the Definition List component is on the same screen, the Definition Attribute List will find the Definition List component and register as a listener. Anytime the sample definition changes or the users selects a different sample definition, the Definition Attribute List the attributes will be updated automatically.

Name	Description	Data Type	Enabled	Required
Total Inspected		Inspected Count	☑	☑
Speck		Nonconforming Co...	☑	☑
Scratch		Nonconforming Co...	☑	☑
Hole		Nonconforming Co...	☑	☑
Discoloration		Nonconforming Co...	☑	☑
Incorrect Size		Nonconforming Co...	☑	☑

Sample Definition Attribute List

The Ignition table customizer is used to change the appearance of the table. To access the customizer, right click on the Definition Attribute List component and select the Cutomizers->Table Customizer menu item. Using the customizer, you can hide columns, change colors, and change formatting to make the Definition Attribute List appear as desired.

When the Read Only property is set to false, the Move Up and Move Down menu items will appear in the popup menu. This allows the user to change the order that attributes appear in the Sample Entry component.

Properties

This component has standard Ignition properties with the addition of the following properties:

© Inductive Automation

Show Disabled When set to true, disabled sample attributes will be shown. This provides a method to re-enable previously disabled sample attributes.

```
Scripting name     showDisabled
Data Type          boolean
```

Attribute Name The attribute name property does not show in the Ignition Designer property list. It is only available in scripting to read the name of the current attribute that is selected when the Edit menu item is clicked.

```
Scripting name     attrName
Data Type          String
```

Read Only When set to true, prevents the popup menu from appearing when the user right-clicks on the Definition Attribute List component.

```
Scripting name     readOnly
Data Type          boolean
```

Events

This component has standard Ignition events with the addition of the following events:

add

Is fired when "Add" menu item is selected. The "Add" menu item will only appear if script has been added to this event.

Event Properties — (none)

edit

Event Properties

event.getSampleAttrName() — Return the currently selected sample definition attribute name.
Data Type: String

remove

Event Properties

event.getSampleAttrName() — Return the currently selected sample definition attribute name.
Data Type: String

event.setRemoveAttribute(boolean) — Used to tell the Definition Attribute List component to remove the selected sample definition attribute. If this is not included in the remove event script with a parameter of 1, then the sample definition attribute will have to be removed using another method.

event.setSuppressConfirmation(boolean) — By including this in the remove event script with a parameter of 1, the confirmation message will not be shown before removing a sample definition attribute. Including the event.RemoveAttribute(0) and setSuppressConfirmation(1) script lines in the remove event will prevent default handling of sample definition

© Inductive Automation

attributes. This allows for custom handling of the removal of sample definition attributes.

Methods

`(none)`

3.4.1.3 Definition Location List

Description

A component that provides a list of production locations that a sample can be taken from for the associated sample definition. In other words, a test is defined (sample definition) and it has locations that are appropriate to take the test at (production location).

There is no need for SQL queries or scripting to display allowable locations. If the Definition List component is on the same screen, the Definition Location List will find the Definition List component and register as a listener. Anytime the sample definition changes or the users selects a different sample definition, the Definition Location List will be updated automatically.

Location Name	Interval Type	Interval	Auto Approve	Enabled
Line 1 Quality	Manual	0	☑	☑
Line 2 Quality	Manual	0	☑	☑

Sample Definition Location List

The Ignition table customizer is used to change the appearance of the table. To access the customizer, right-click on the Definition Location List component and select the Cutomizers->Table Customizer menu item. Using the customizer, you can hide columns, change colors, and change formatting to make the Definition Location List appear as desired.

When the Read Only property is set to false, the Move Up and Move Down menu items will appear in the popup menu. This allows the user to change the order that attributes appear in the Sample Entry component.

Properties

This component has standard Ignition properties with the addition of the following properties:

© Inductive Automation

Location ID The location ID property does not show in the Ignition Designer property list. It on only available in scripting to read the ID of the current production location that is selected when the Edit menu item is clicked.

```
Scripting name     locID
Data Type          int
```

Read Only When set to true, prevents the popup menu from appearing when the user right-clicks on the Definition Location List component.

```
Scripting name     readOnly
Data Type          boolean
```

Events

This component has standard Ignition events with the addition of the following events:

add

Is fired when "Add" menu item is selected. The "Add" menu item will only appear if script has been added to this event.

Event Properties: (none)

edit

Event Properties

event.getSampleLocName()	Return the currently selected sample definition location name. Data Type: String

remove

Event Properties

event.getSampleLocName()	Return the currently selected sample definition location name. Data Type: String
event.setRemoveLocation(boolean)	Used to tell the Definition Location List component to remove the selected sample definition location. If this is not included in the remove event script with a parameter of 1, then the sample definition location will have to be removed using another method.
event.setSuppressConfirmation(boolean)	By including this in the remove event script with a parameter of 1, the confirmation message will not be shown before removing a sample definition location. Including the event.RemoveLocation(0) and setSuppressConfirmation(1) script lines in the remove event will prevent default handling of sample definition locations. This allows for custom handling of the removal of sample definition locations.

Methods

© Inductive Automation

(none)

3.4.1.4 Definition Control Limit List

Description

A component that provides a list of control limits to apply to a sample definition. All control limits that are configured in the project will appear in the list and can be selected by the user. Control limits that are selected by the user will be available to show on control charts and may be used during automatic signal evaluation.

There is no need for SQL queries or scripting to display control limits. If the Definition List component is on the same screen, the Definition Control Limit List will find the Definition List component and register as a listener. Anytime the sample definition changes or the users selects a different sample definition, the Definition Control Limit List will be updated automatically.

Name	Enable
Histogram LCL	☐
Histogram LCL	☐
Histogram UCL	☐
Histogram UCL	☐
Individual LCL	☐
Individual LCL	☐
Individual UCL	☐

Sample Definition Control Limit List

Properties

This component has standard Ignition properties.

(none)

Events

This component has standard Ignition events.

(none)

Methods

(none)

3.4.1.5 Definition Signals List

Description

A component that provides a list of signals (rules) to apply to a sample definition. All signals that are configured in the project will appear in the list and can be selected by the user. Signals that are selected by the user will be available to show on control charts and will be automatically evaluated when new samples are added.

There is no need for SQL queries or scripting to display signals. If the Definition List component is on the same screen, the Definition Signals List will find the Definition List component and register as a listener. Anytime the sample definition changes or the users selects a different sample definition, the Definition Signals List will be updated automatically.

Name	Enable
Individual Outside	☐
Individual Outside	☐
Out of Limits	☐
Out of Limits	☐
Outside Limits	☐
Outside Limits	☐
XBar 8 Above Control Line	☐

Sample Definition Signal List

Properties

This component has standard Ignition properties.

(none)

Events

This component has standard Ignition events.

(none)

Methods

(none)

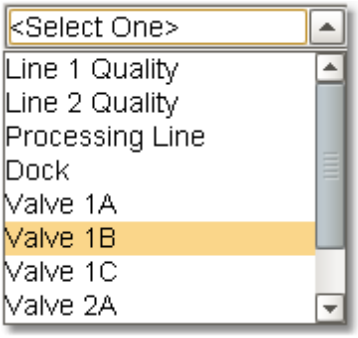
3.4.1.6 Location Selector

Description

A component that allows selection of production locations. Production locations are defined in the

production model using the Ignition Designer. See Production Model Configuration for more information.

There is no need for SQL queries or scripting to display locations. The selected location is reflected in Selected Location Name, Path and Location ID properties.



Location Selector

Properties

This component has standard Ignition properties with the addition of the following properties:

Selected Location Name

The name of the currently selected location.

```
Scripting name  selectedLocationName
Data Type  String
```

Selected Location Path

The full location path of the currently selected location. This includes the project, enterprise, site, area, and possibly a line and the location, each separated by the backslash \ character. Example: QualityDemo\New Enterprise\New Site\Packaging\Line 1\Line 1 Quality

```
Scripting name     selectedLocationPath
Data Type          String
```

Selected Path Without Project

The location path, excluding the project name, of the currently selected location. This is useful when using this component with the SPCController component. This includes the enterprise, site, area, possible a line and the location each separated by the backslash \ character. Example: New Enterprise\New Site\Packaging\Line 1\Line 1 Quality

```
Scripting name     selectedPathWithoutProject
Data Type          String
```

Selected Location ID

The location ID of the currently selected location.

```
Scripting name     selectedLocationID
Data Type          int
```

Display Path

When set to true, the full location paths will be displayed in the drop down list. Otherwise, just the location name will be displayed.

```
Scripting name     displayPath
Data Type          boolean
```

© Inductive Automation

Events

This component has standard Ignition events.

(none)

Methods

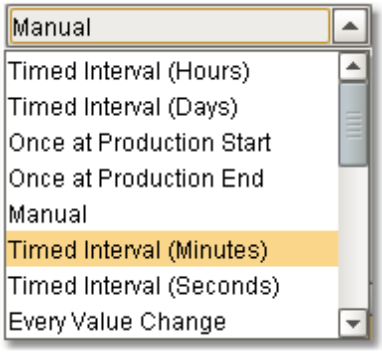
(none)

3.4.1.7 Interval Selector

Description

A component that allows selection of sample intervals. All intervals that are configured in the project will appear in the list and can be selected by the user. See the Sample Intervals section for more information on intervals.

There is no need for SQL queries or scripting to display intervals.



Interval Selector

Properties

This component has standard Ignition properties with the addition of the following properties:

Selected Interval The name of the currently selected interval.

```
Scripting name     selectedInterval
Data Type          String
```

Events

This component has standard Ignition events.

(none)

Methods

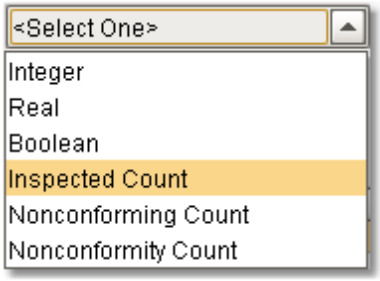
(none)

3.4.1.8 Datatype Selector

Description

A component that allows selection of sample attribute data types. The data types are built into the SPC module and cannot be added to or changed.

There is no need for SQL queries or scripting to display the data types.



Data Type Selector

The following table describes each data type.

Data Type	Description	Range
Integer	Positive and negative numbers without decimal points and fractional digits.	-2,147,483,648 to 2,147,483,647
Real	Numbers including decimal points and fractional digits.	1.40129846432481707e-45 to 3.40282346638528860e+38 (positive or negative)
Boolean	True or false	True or false
Inspected Count	A count of inspected units in an integer format. This is used for attribute types of sample definitions.	up to 2,147,483,647
Nonconforming Count	A count of nonconforming (defective) units in an integer format. This is used for attribute types of sample definitions.	up to 2,147,483,647
Nonconformity Count	A count of nonconformities (defects) in an integer format. This is used for attribute types of sample definitions.	up to 2,147,483,647

Properties

This component has standard Ignition properties with the addition of the following properties:

Selected Data Type The currently selected data type.

```
Scripting name     selectedDataType
Data Type          AttributeDataType
```

© Inductive Automation

Events

This component has standard Ignition events.

(none)

Methods

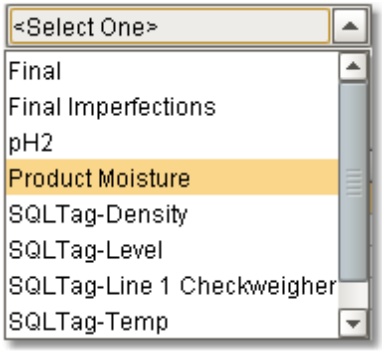
(none)

3.4.1.9 Definition Selector

Description

A component that allows selection of sample definitions. One source of sample definitions is from the definition management screen that uses the Definition List component.

There is no need for SQL queries or scripting to display the data types.



Sample Definition Selector

When an allowable location is added to a sample definition, a tag value can be set. This component can limit the sample definitions that appear by entering in a matching tag values. It is typically used for defining who has ownership for collecting sample data. For example, the lab takes samples at packaging line 1 every 2 hours. The operator also takes samples at packaging line 1 every 1 hour. When the lab takes a sample, they don't want to see information that the operator has ownership for and visa versa. To accomplish this, set the tag value to "Lab" for sample definitions that the lab has ownership for and to "Operator" for sample definitions that the operator has ownership for.

Properties

This component has standard Ignition properties with the addition of the following properties:

© Inductive Automation

Location Path
Set to a valid path of a production location item to show the sample definition for the location.

```
Scripting name     locationPath
Data Type          String
```

Tag
Optionally, set to a value to filter the sample definition by.

```
Scripting name     locationPath
Data Type          String
```

Selected Sample Definition Name
The currently selected sample definition name.

```
Scripting name     selectedSampleDefinitionName
Data Type          String
```

Selected Sample DefUUID
Return the UUID assigned to the currently selected sample definition. A UUID is a universally unique identifier that, once assigned to a sample definition, will never change. It is automatically generated when a sample definition is created and is unique in that no two samples definitions will have the same UUID.

```
Scripting name     selectedSampleDefUUID
Data Type          String
```

Events

This component has standard Ignition events.

(none)

Methods

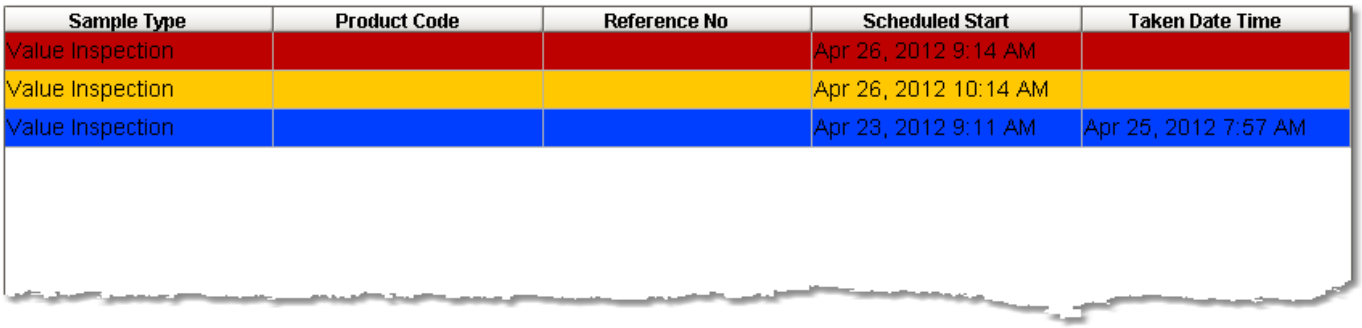
(none)

3.4.1.10 Location Sample List

Description

A component that displays samples for a location and optionally by sample ownership. Through configuration properties, it can show samples that are scheduled to be coming due, due, overdue, or waiting approval or approved.

There is no need for SQL queries or scripting to display the samples.



Sample Type	Product Code	Reference No	Scheduled Start	Taken Date Time
Value Inspection			Apr 26, 2012 9:14 AM	
Value Inspection			Apr 26, 2012 10:14 AM	
Value Inspection			Apr 23, 2012 9:11 AM	Apr 25, 2012 7:57 AM

Sample Definition Selector

When an allowable location is added to a sample definition, a tag value can be set. This component can limit the samples that appear by entering in matching tag values. It is typically used for defining who has ownership for collecting sample data. For example, the lab takes samples at packaging line 1 every 2 hours. The operator also takes samples at packaging line 1 every 1 hour. The lab does not want to see samples that the operator has ownership for and vice versa. To accomplish this, set the tag value to "Lab" for sample definitions that the lab has ownership for and to "Operator" for sample definitions that the operator has ownership for.

The Ignition table customizer is used to change the appearance of the table. To access the customizer, right-click on the Location Sample List component and select the Cutomizers->Table Customizer menu item. Using the customizer, you can hide columns, change colors, change formatting to make the Location Sample List appear as desired.

Properties

This component has standard Ignition properties with the addition of the following properties:

Location Path

Set to a valid path of a production location item to show samples for that location.

```
Scripting name     locationPath
Data Type          String
```

Tag

Optionally, set to a value to filter the samples by.

```
Scripting name     locationPath
Data Type          String
```

Read Only

When set to true, prevents the popup menu from appearing when the user right-clicks on the Location Sample List component.

```
Scripting name     readOnly
Data Type          boolean
```

Show Waiting Approval

When set to true, includes samples that are waiting approval.

```
Scripting name     showWaitingApproval
Data Type          boolean
```

Show Approved Samples

When set to true, includes samples that have been approved.

```
Scripting name     showApprovedSamples
Data Type          boolean
```

Show Due Samples

When set to true, includes samples that are due.

```
Scripting name     showDueSamples
Data Type          boolean
```

© Inductive Automation

Show Coming Due Samples When set to true, includes samples that are coming due.
Scripting name showDueSamples
Data Type boolean

Show Overdue Samples When set to true, includes samples that are overdue.
Scripting name showOverdueSamples
Data Type boolean

Show Removed Samples When set to true, includes samples that have been previously removed.
Scripting name showRemovedSamples
Data Type boolean

Sort Type Changes the order that the sample will appear in the list.
Options:

None	Display samples in natural order.
Due State	Display samples in order of the severity of due sta
Taken Date Time	Display samples in order by the date it is taken.
Taken Date Time (Descending)	Display samples in reverse order by the date it is

Scripting name sortType
Data Type int
Numeric value used in scripting.

None	0
Due State	1
Taken Date Time	2
Taken Date Time (Descending)	3

Enable Note Editing When set to true, allows users to enter a note tied to a sample.
Scripting name showEnableNoteEditing
Data Type boolean

Start Date Optionally, set this property to only show samples that are scheduled after the specified Start Date.
Scripting name startDate
Data Type Date

End Date Optionally, set this property to only show samples that are scheduled before the specified End Date.
Scripting name endDate
Data Type Date

Product Code Optionally, set this property to only show samples for specified product code.
Scripting name productCode
Data Type String

Reference No Optionally, set this property to only show samples for specified reference number. Reference numbers can represent anything such as lot number, batch number, raw material lot number, raw material vendor, etc.
Scripting name referenceNo
Data Type String

Events

This component has standard Ignition events with the addition of the following events:

add
Is fired when "Add" menu item is selected. The "Add" menu item will only appear if script has been added to this event.

Event Properties
(none)

edit
Is fired when "Edit" menu item is selected. The "Edit" menu item will only appear if script has been added to this event.

Event Properties

event.getSampleUUID()
Return the sample UUID for the currently selected sample. A UUID is a universally unique identifier that, once assigned to a sample, will never change. It is automatically generated when a sample is created and is unique in that no two samples will have the same UUID.
Data Type String

remove
Is fired when "Remove" menu item is selected. The "Remove" menu item will only appear if script has been added to this event.

Event Properties

event.getSampleUUID()
Return the sample UUID for the currently selected sample. A UUID is a universally unique identifier that, once assigned to a sample, will never change. It is automatically generated when a sample is created and is unique in that no two samples will have the same UUID.
Data Type String

approve
Is fired when "Approve" menu item is selected. The "Approve" menu item will only appear if script has been added to this event.

Event Properties

event.getSampleUUID()
Return the sample UUID for the currently selected sample. A UUID is a universally unique identifier that, once assigned to a sample, will never change. It is automatically generated when a sample is created and is unique in that no two samples will have the same UUID.
Data Type String

unapprove
Is fired when "Unapprove" menu item is selected. The "Unapprove" menu item will only appear if script has been added to this event.

Event Properties

event.getSampleUUID()
Return the sample UUID for the currently selected sample. A UUID is a universally unique identifier that, once assigned to a sample, will never change. It is automatically generated when a sample is created and is unique in that no two samples will have the same UUID.
Data Type String

review
Is fired when "Review" menu item is selected. The "Review" menu item will only appear if script has been added to this event.

Event Properties

event.getSampleUUID()
Return the sample UUID for the currently selected sample. A UUID is a universally unique identifier that, once assigned to a sample, will never change. It is automatically generated when a sample is created and is unique in that no two samples will have the same UUID.
Data Type String

Methods

© Inductive Automation

`createByDefUUID(defUUID)`

Create a new sample based on the sample definition specified by the defUUID parameter.

parameters

`defUUID` Sample definition UUID to base the new sample on. A UUID is a universally unique identifier that, once assigned to a sample definition, will never change. It is automatically generated when a sample definition is created and is unique in that no two samples definitions will have the same UUID.
`Data Type` String

See SampleDefinition Object for more information.

returns

Sample An instance of a new sample
`Data Type` Sample

See Sample Object for more information.

`createByDefName(defName)`

Create a new sample based on the sample definition specified by the defName parameter.

parameters

`defName` Sample definition name to base the new sample on.
`Data Type` String

See SampleDefinition Object for more information.

returns

Sample An instance of a new sample
`Data Type` Sample

See Sample Object for more information.

`update(sample)`

Create a new sample based on the sample definition specified by the defName parameter.

parameters

`sample` This is the sample to either update, if it already exists, or add, if it does not already exist.
`Data Type` Sample

See Sample Object for more information.

returns

String Message of any errors that may have occurred during the update operation.
`Data Type` String

`approve(sample)`

Approve the sample specified by the sample parameter.

parameters

`sample` This is the sample to approve.
`Data Type` Sample

See Sample Object for more information.

returns

String Message of any errors that may have occurred during the approve operation.
`Data Type` String

© Inductive Automation

`unapprove(sample)`

Unapprove the sample specified by the sample parameter.

parameters

`sample` This is the sample to unapprove.
`Data Type` Sample
See Sample Object for more information.

returns

String Message of any errors that may have occurred during the unapprove operation.
`Data Type` String

`remove(sample)`

Remove the sample specified by the sample parameter. **Caution: this will permanently remove the data from the database and it cannot be recovered.**

parameters

`sample` This is the sample to remove.
`Data Type` Sample
See Sample Object for more information.

returns

String Message of any errors that may have occurred during the remove operation.
`Data Type` String

`getSample(sampleUUID)`

Return the sample specified by the sampleUUID parameter.

parameters

`sampleUUID` Sample UUID to return the sample for. A UUID is a universally unique identifier that, once assigned to a sample, will never change. It is automatically generated when a sample is created and is unique in that no two samples will have the same UUID.
`Data Type` String
See Sample Object for more information.

returns

Sample An instance of a sample
`Data Type` Sample
See Sample Object for more information.

`showEditNotePopup()`

Show the note popup to allow the user to add or edit the note tied to the currently selected sample.

parameters (none)
returns nothing

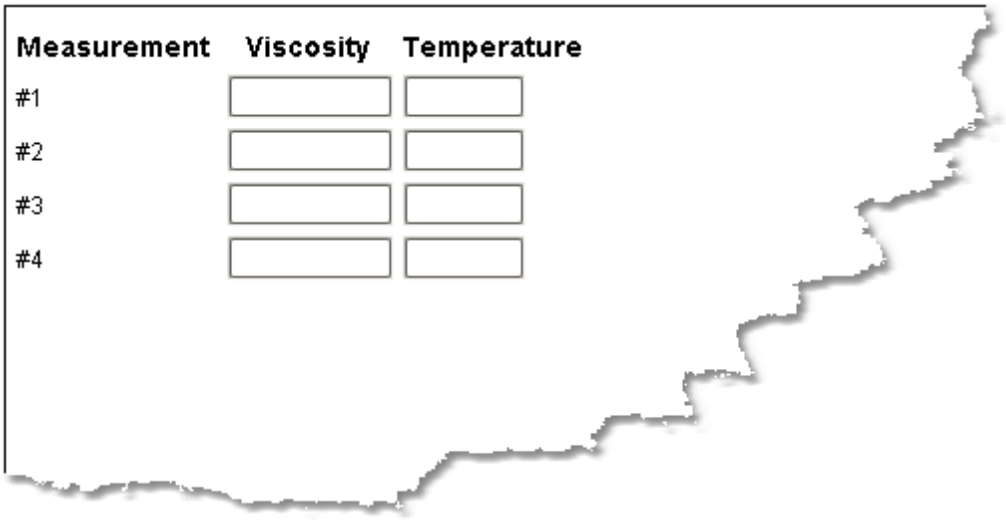
© Inductive Automation

3.4.1.11 Sample Entry

Description

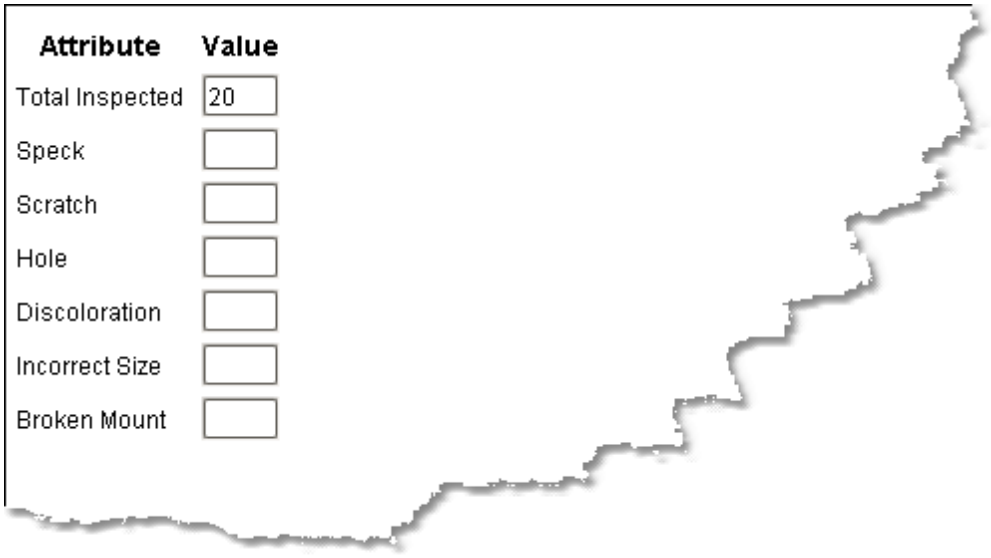
A component used to display and enter sample measurement data. The entry fields are dynamically created based on attributes defined in the sample definition. Additionally, the number of measurements are defined by the measurement count setting in the sample definition. The Up Down Traversal property can be used to change the field tab order between column and row. When saving, the measurement data is validated, and if any validation errors exists a message is displayed to the user.

Depending on the measurement count defined in the sample definition, the orientation of the edit fields will change. If the measurement count is greater than 1, then there will be a row for each measurement with the attributes appearing horizontally. If the measurement count is equal to 1, then the attributes appear vertically in separate rows. This reduces the need for the user to have to scroll while entering sample data if the are a number of attributes.



Multiple Measurement Sample Entry

© Inductive Automation



Single Measurement Sample Entry

Properties

This component has standard Ignition properties with the addition of the following properties:

Up Down Traversal

When set to true, causes the focused field to move down to the next field when the Tab or Enter keys are pressed. If it is on the last measurement, it will move to the top field in the next column. When set to false, causes the focused field to move right to the next field when the Tab or Enter keys are presses. If it is on the last column, it will move to the left column of the next row.

```
Scripting name     upDownTraversal
Data Type          boolean
```

Read Only

When set to true, prevents the popup menu from appearing when the user right-clicks on the Location Sample List component.

```
Scripting name     readOnly
Data Type          boolean
```

Sample Taken Date Time

When set, it will be used for the date and time the sample was taken. If not set, the current date and time will be used for the date and time the sample was taken. This is useful if samples are taken but not entered until a later time.

```
Scripting name     sampleTakenDateTime
Data Type          Date
```

Foreground Color

This is the color of the text within the Sample Entry component.

```
Scripting name     foregroundColor
Data Type          Color
```

Background Color

This is the color of the body of the Sample Entry component.

```
Scripting name     backgroundColor
Data Type          Color
```

Measurement Label

This is the header text for the measurement column.

```
Scripting name     measurementLabel
Data Type          String
```

© Inductive Automation

Label Font Font used for the column headers.
Scripting name labelFont
Data Type Font

Title Text to show at the top of the Sample Entry component.
Scripting name title
Data Type String

Title Font Font to use for the title.
Scripting name titleFont
Data Type Font

Measurement Number Font Font to use for the measurement numbers.
Scripting name numberFont
Data Type Font

Entry Field Font Font to use for the entry fields.
Scripting name fieldFont
Data Type Font

Column Gap Size Space in pixels between columns.
Scripting name gapx
Data Type int

Row Gap Size Space in pixels between rows.
Scripting name gapy
Data Type int

Events

This component has standard Ignition events.

(none)

Methods

`save()`

Save changes made to the measurement values. This method also records the current product code and reference number for the production location.

parameters (none)

returns

String Message of any errors that may have occurred during the save operation.
Data Type String

`save(productCode, refNo)`

Save changes made to the measurements values along with a product code and reference number specified in the parameters.

parameters

© Inductive Automation

productCode — Product code to record along with the measurement values.
Data Type String
refNo — Reference number to record along with the measurement values.
Data Type String

returns

String — Message of any errors that may have occurred during the save operation.
Data Type String

`undo()`

Any changed measurement values will be restored to their original values.

parameters (none)
returns nothing

`approve()`

Approve the current sample.

parameters (none)
returns

String — Message of any errors that may have occurred during the approve operation.
Data Type String

`unapprove()`

Unapprove the current sample.

parameters (none)
returns

String — Message of any errors that may have occurred during the unapprove operation.
Data Type String

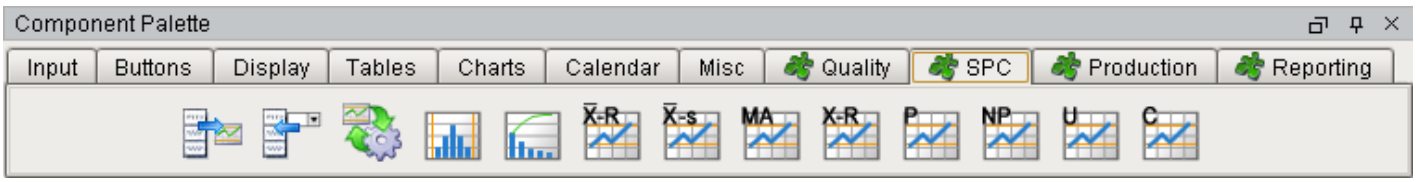
`showEditNotePopup()`

Show the note popup to allow the user to add or edit the note tied to the currently selected sample.

parameters (none)
returns

String — Message of any errors that may have occurred during the show note operation.
Data Type String

3.4.2 SPC Components



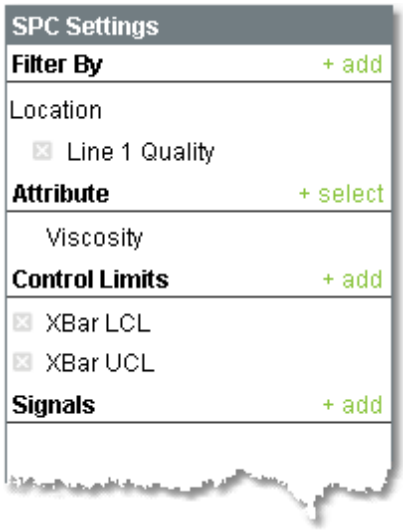
SPC Components

© Inductive Automation

3.4.2.1 SPC Selector

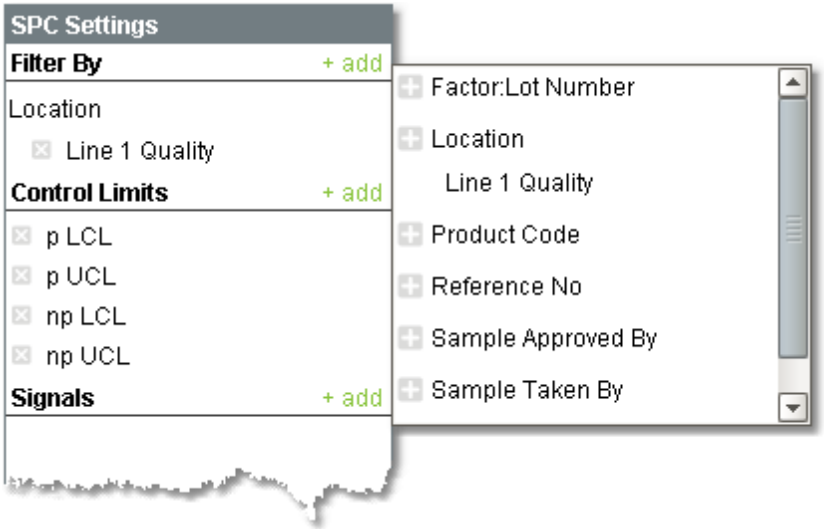
Description

A component that allows selections of SPC data. As the user makes selections, this component will query the server for results. These results can be accessed through the SPC Results and SPC Data and can be linked with any of the SPC control charts.



SPC Selector

A filter can be added by selecting the + add link to the right of **Filter By**. A window panel will open and filter categories will be displayed. Click the ⊞ link by the filter category and specific filter items will be displayed. When selected they will be added to the filters as shown below. To minimize the number of filter options, reduce the date range defined by the Start Date and End Date properties and the associated filter values will be shown. Because values collected from different locations being shown together does not make sense, a location must be added to the Filter By section.

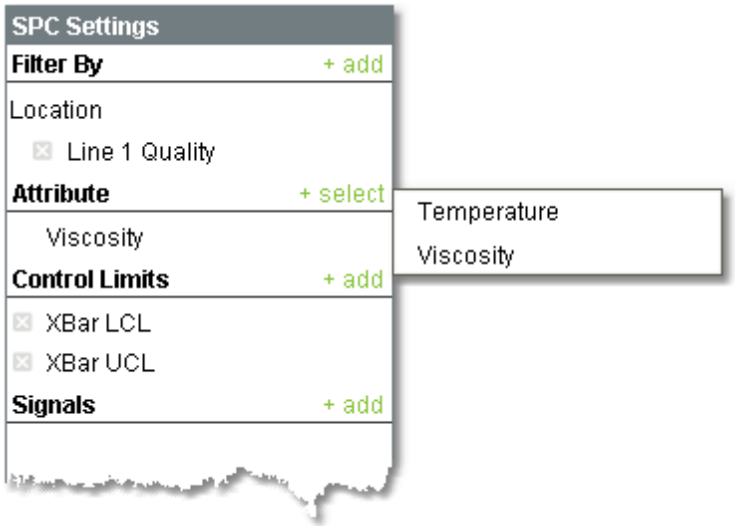


Filter By List

© Inductive Automation

Sample definitions can have more than one attribute. At the time sample data is recorded, each attribute will have a value associated with it. For example, when collecting viscosity reading it may also be important to know the temperature. But, showing and making calculations on a viscosity value of 10000 with a temperature value of 75.2 does not make sense. The SPC Selector allow selecting a single attribute as shown below.

If a attribute type of sample definition is selected, then the Attribute section will not appear. This is because with attribute charts, all attributes are included and shown. For example, if a sample definition has an attribute for Torn, Discolored, Pitted, etc. then all will show in the table and included in the calculations.



Attribute Selection

Similar to filters, control limits and signals can be added to the SPC results. Any selected control limits, and signals that depend on them, will not appear on the control chart until the control limit value has been set.

Selections can be removed by selecting the ⊠ link to the left of the selection.

To display the SPC results of this component in a control charts, bind the SPC Results property of the control chart to the SPC Results property of this component.

Properties

This component has standard Ignition properties with the addition of the following properties:

Start Date

This property is the starting date for retrieving analysis data and determining available filter and compare by options.

```
Scripting name        startDate
Data Type             Date
```

End Date

This property is the ending date for retrieving analysis data and determining available filter and compare by options.

```
Scripting name        endDate
Data Type             Date
```

© Inductive Automation

Definition Name The sample definition to used when building SPC results.

```
Scripting name        definitionName
Data Type             String
```

Auto Refresh If true, the SPC results will be updated every time a new sample is added for the selected sample definition and location.

```
Scripting name        autoRefresh
Data Type             Boolean
```

SPC Results This bind only property holds the SPC results and includes a data set with the raw data, sample definition information and calculated value information. With all the information included in the SPC Results, control charts can display the results which is not possible with the data set alone.

```
Scripting name        spcResults
Data Type             SPCResults
```

SPC Data This property holds the SPC data and includes columns for the calculated values based on the SPC Data Format and selected attribute, control limits and signals.

```
Scripting name        spcData
Data Type             Dataset
```

Suppress Warnings If true, any warnings received back when requesting SPC results are not shown. Instead the Message property can be bound to display any warning in a text or other component. Warning include requesting SPC data with settings that do not make sense. For example, requesting p chart results for a sample definition that contains no attribute type of data.

```
Scripting name        suppressWarnings
Data Type             Boolean
```

Suppress Errors If true, any errors received back when requesting SPC results are not shown.

```
Scripting name        suppressErrors
Data Type             Boolean
```

Include Disabled Attributes If true, any attributes that have been disabled in the sample definition will appear in the attribute selection panel. This provides a method to view old attribute data that has been disabled.

```
Scripting name        includeDisabledAttributes
Data Type             Boolean
```

Filter Selection Summary This property holds the current filter item selections that the results will be filtered by. If more than one item exists, they are separated by commas.

```
Scripting name        filterSummary
Data Type             String
```

Attribute Name

This property holds the currently selected attribute to include in the results. For each sample definition there may be multiple attributes that are collected. This property selects which one to show get the SPC Results for.

```
Scripting name        attributeName
Data Type             String
```

Control Limit Summary

This property holds the current control limit selections to include in the results. If more than one item exists, they are separated by commas.

```
Scripting name        controlLimitSummary
Data Type             String
```

Signal Summary

This property holds the currently selected signals to include in the results. If more than one item exists, they are separated by commas.

```
Scripting name        signalSummary
Data Type             String
```

SPC Data Format

This property specifies the type of control chart to retrieve the SPC data for.

Options:

None - No results will be returned.
XBarR - XBar and range data will be returned.
XBarS - XBar and standard deviation data will be returned.
Individual - Individual and moving range data will be returned.
Median - Median and moving range data will be returned.
P - P chart data will be returned.
NP - NP chart data will be returned.
C - C chart data will be returned.
U - U chart data will be returned.
Histogram - Histogram data will be returned.
Pareto - Pareto data will be returned.

```
Scripting name        spcDataFormat
Data Type             SPCDataFormat
```

Numeric value used in scripting.

None	0
XBarR	1
XBarS	2
Individual	3
Median	4
U	5
C	6
P	7
NP	8
Histogram	9
Pareto	10

Auto Bar Count

If set to true, the number of histogram bars will be automatically determined.

```
Scripting name        autoBarCount
Data Type             boolean
```

© Inductive Automation

Data Bar Count If Auto Bar Count is set to false, the value of this property will determine the number of histogram bars.

```
Scripting name        dataBarCount
Data Type             int
```

Padding Bar Count The value of this property determines how many empty bars will be included in histogram results.

```
Scripting name        paddingBarCount
Data Type             int
```

Events

This component has standard Ignition events.

Methods

`refreshInfo()`

Force refresh of the SPC results.

parameters (none)

returns nothing

`setSpcDataFormat(spcDataFormat)`

Change to format if the SPC data to return.

parameters

spcDataFormat — Format of the SPC data to return.

Data Type	int
None	0
XBarR	1
XBarS	2
Individual	3
Median	4
U	5
C	6
P	7
NP	8
Histogram	9
Pareto	10

returns nothing

`setRowLimit(rowLimit)`

Change the default number of samples to return to the value specified in the rowLimit parameter. By default only 500 samples are returned in the SPC results. This is done to unburden the database, network bandwidth and memory.

parameters

rowLimit — New row limit.

```
Data Type             int
```

returns nothing

`getRowLimit()`

Returns the current row limit value.

parameters (none)
returns nothing

Example Code

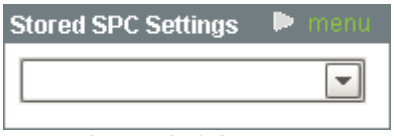
This script will change the format of the SPC results.

```
event.source.parent.getComponent('SPC Selector').setSpcDataFormat(system.quality.spc.format.P.getValue())
```

3.4.2.2 Stored SPC Selector

Description

A component that allows creating, recalling and saving SPC selections in the SPC Selector component. This component will automatically use the available SPC Selector in the container. Keep in mind that whenever a new sample definition is created, a new stored SPC settings items will be created with the default values. This being said, additional stored SPC settings items can be created each with different filters, attribute, control limits and signals.

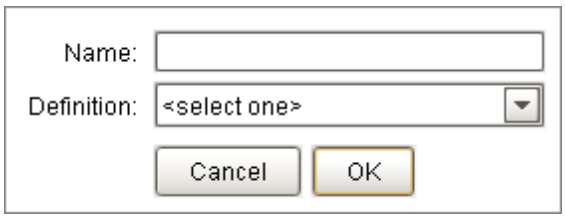


Stored SPC Selector

By clicking on the

link, a menu with the option to create new, save, delete and rename SPC settings will popup.

To add a new saved SPC settings item, click on **New** menu item, enter a name, select a sample definition and click **OK**. This will create a default SPC Settings item. Now the user can select filters, attribute, control limits and signals that will be saved and can easily be selected at a later time.



New Stored SPC Settings

To rename a stored SPC Settings item, select an item and click on the **Rename** menu item, enter a new name and click **OK**.

Rename Stored SPC Settings

To delete a stored SPC Settings item, select an item and click on **Delete** menu item, and select **Yes** to

the confirmation message.

If changes to a stored SPC settings values have been made and the user selects a different stored SPC Settings, they will be prompted to save the changes. Alternatively, the changes can be saved by clicking on the **Save** menu item.

Properties

This component has standard Ignition properties with the addition of the following properties:

Show Disabled When true, disabled attributes will be included in the attribute list.

```
Scripting name        showDisabled
Data Type             Boolean
```

Show Menu When true, the menu will be displayed. By setting the property to false, it will only allow users to select stored SPC settings items and prevent them from creating new, renaming existing, saving over existing or deleting stored SPC Settings items.

```
Scripting name        showMenu
Data Type             Boolean
```

Menu Top The y coordinate to display the menu at.

```
Scripting name        menuTopPosition
Data Type             int
```

Menu Left The x coordinate to display the menu at.

```
Scripting name        menuLeftPosition
Data Type             int
```

Menu Image The image to show for the menu.

```
Scripting name        menuImage
Data Type             Image
```

Events

This component has standard Ignition events with the addition of the following events:

selected — Is fired when a different SPC Settings item is selected menu item is selected.

Event Properties

event.getSettingsName()	Returns the name of the newly selected SPC Settings item.
event.getSettings()	Returns a reference to the SPCSettings object that contains the filter, attribute, control limit and signal selections.

created — Is fired when a new SPC Settings item is created.

Event Properties

event.getSettingsName()	Returns the name of the newly created SPC Settings item.
event.getSettings()	Returns a reference to the SPCSettings object that contains the filter, attribute, control limit and signal selections.

deleted — Is fired when a SPC Settings item is deleted.

Event Properties

event.getSettingsName()	Returns the name of the deleted SPC Settings item.

renamed — Is fired when a SPC Settings item is renamed.

Event Properties

event.getSettingsName()	Returns the new name of the SPC Settings item.
event.getPrevName()	Returns the previous name of the SPC Settings item.
event.getSettings()	Returns a reference to the SPCSettings object that contains the filter, attribute, control limit and signal selections.

Methods

`(none)`

3.4.2.3 SPC Controller

Description

An invisible component that makes SPC data available for reports and other components. The term *invisible component* means that this component appears during design time, but is not visible during runtime.

In cases where the SPC Selector offers too many options to the use, this component can be used. It has all of the same functionality as the SPC Selector but without the user interface. This means property bindings or script must be used to make the filter, compare by and data point selections. It also is used for providing data to canned reports and optionally allowing the user to make limited filter options.

To display the SPC results of this component in a control charts, bind the SPC Results property of the control chart to the SPC Results property of this component.

Properties

This component has standard Ignition properties with the addition of the following properties:

© Inductive Automation

Automatic Update

When true, when any property that changes the results, the results will automatically be updated.

```
Scripting name      automaticUpdate
Data Type           Boolean
```

Auto Refresh

If true, the SPC results will be updated every time a new sample is added for the selected sample definition and location.

```
Scripting name      autoRefresh
Data Type           Boolean
```

Row Limit

The number of samples to return in the SPC Results. This is done to unburden the database, network bandwidth and memory.

```
Scripting name      autoRefresh
Data Type           Boolean
```

SPC Data Format

This property specifies the type of control chart to retrieve the SPC data for.

Options:

None - No results will be returned.
XBarR - XBar and range data will be returned.
XBarS - XBar and standard deviation data will be returned.
Individual - Individual and moving range data will be returned.
Median - Median and moving range data will be returned.
P - P chart data will be returned.
NP - NP chart data will be returned.
C - C chart data will be returned.
U - U chart data will be returned.
Histogram - Histogram data will be returned.
Pareto - Pareto data will be returned.

```
Scripting name      spcDataFormat
Data Type           SPCDataFormat
```

Numeric value used in scripting.

None	0
XBarR	1
XBarS	2
Individual	3
Median	4
U	5
C	6
P	7
NP	8
Histogram	9
Pareto	10

Start Date

This property is the starting date for retrieving analysis data and determining available filter and compare by options.

```
Scripting name      startDate
Data Type           Date
```

End Date

This property is the ending date for retrieving analysis data and determining available filter and compare by options.

```
Scripting name        endDate
Data Type             Date
```

Stored SPC Name

This optional property can be used to populate the Definition Name, Attribute Name, FIlter, Control Limits, SIgnals, SPC Data Format properties with those in a store SPC settings. Stored SPC settings are saved using the SPC Selector and Stored SPC Selector components.

```
Scripting name        storedSPCName
Data Type             String
```

Definition Name

The sample definition to used when building SPC results.

```
Scripting name        definitionName
Data Type             String
```

Attribute Name

This property holds the attribute from the sample definition to include in the results. For each sample definition there may be multiple attributes that are collected. This property selects which one to show get the SPC Results for.

```
Scripting name        attributeName
Data Type             String
```

Filter

This property holds the filter expression that the results will be filtered by. If more than one item exists, they are separated by commas. Example: Approved By=John Doe

```
Scripting name        filter
Data Type             String
```

Control Limits

This property holds the control limits to include in the results. If more than one item exists, they are separated by commas. Example: Individual LCL, Individual UCL

```
Scripting name        controlLimits
Data Type             String
```

Signals

This property holds the signals to include in the results. If more than one item exists, they are separated by commas.
Example: Individual Outside

```
Scripting name        signals
Data Type             String
```

SPC Results

This bind only property holds the SPC results and includes a data set with the raw data, sample definition information and calculated value information. With all the information included in the SPC Results, control charts can display the results which is not possible with the data set alone.

```
Scripting name        spcResults
Data Type             SPCResults
```

© Inductive Automation

SPC Data This property holds the SPC data and includes columns for the calculated values based on the SPC Data Format and selected attribute, control limits and signals.

```
Scripting name        spcData
Data Type             Dataset
```

Include Disabled Attributes If true, any attributes that have been disabled in the sample definition will appear in the attribute selection panel. This provides a method to view old attribute data that has been disabled.

```
Scripting name        includeDisabledAttributes
Data Type             Boolean
```

Error Message If an error is encountered while retrieving SPC results, it will be readable from this property.

```
Scripting name        errorMessage
Data Type             String
```

Warning Message If a warning is encountered while retrieving SPC results, it will be readable from this property.

```
Scripting name        warningMessage
Data Type             String
```

Auto Bar Count If set to true, the number of histogram bars will be automatically determined.

```
Scripting name        autoBarCount
Data Type             boolean
```

Data Bar Count If Auto Bar Count is set to false, the value of this property will determine the number of histogram bars.

```
Scripting name        dataBarCount
Data Type             int
```

Padding Bar Count The value of this property determines how many empty bars will be included in histogram results.

```
Scripting name        paddingBarCount
Data Type             int
```

Dynamic Properties Depending on the setting of the Definition Name property, the dynamic properties will change. A dynamic property to be created for each filter category that can be bound to by other components. These dynamic properties can also be set through script.

Events

This component has standard Ignition events with the addition of the following events:

beforeUpdate Is fired just before SPC results are requested from the SPC module.
Event Properties (none)

afterUpdate Is fired just after SPC results are requested from the SPC module.
Event Properties (none)

© Inductive Automation

Methods

`refreshInfo()`

Causes the sample definition information to be refreshed.

parameters	(none)
returns	nothing

`update()`

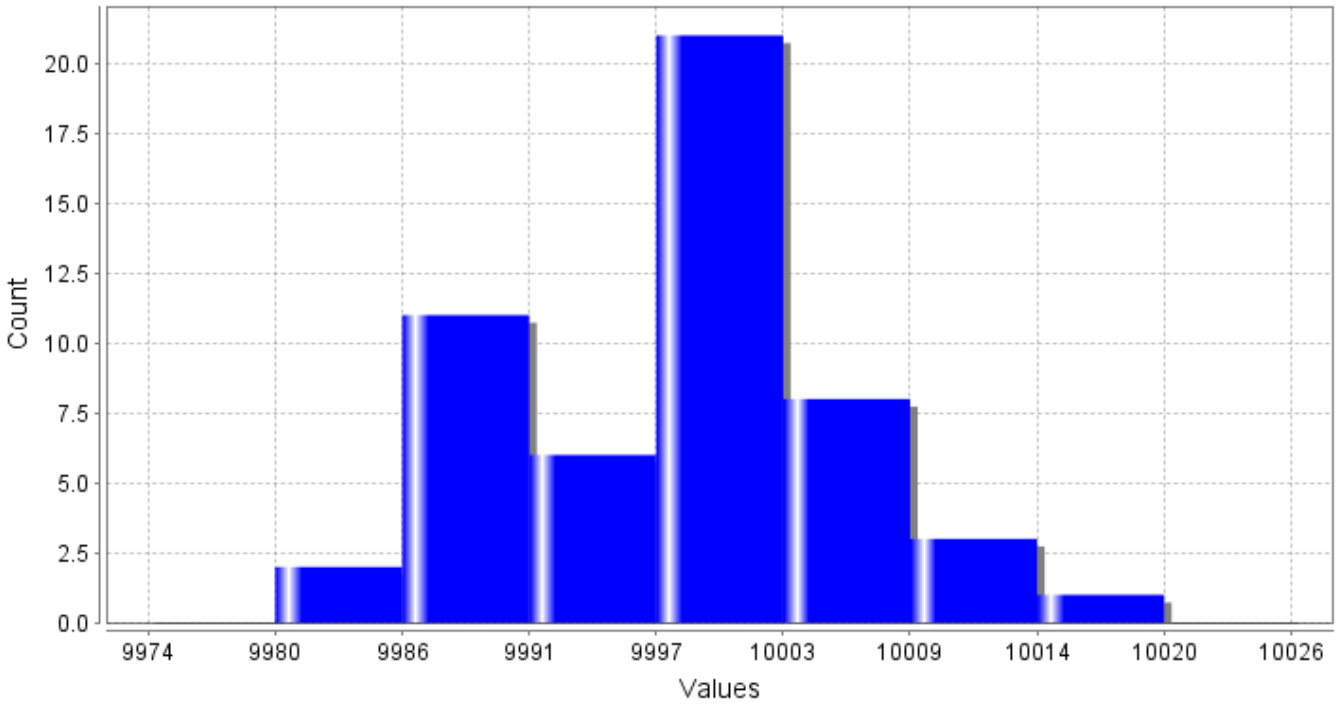
Causes the SPC results to be updated.

parameters	(none)
returns	nothing

3.4.2.4 Histogram Chart

Description

The Histogram chart is used to display frequency distribution of sample measurements. It does not retrieve SPC results from the SPC module so it must be used with either the SPC Selector or the SPC Controller components that do. Only SPC results with Histogram SPC Data Format specified will be displayed.



Histogram Chart

© Inductive Automation

Through the use of the properties listed below, the appearance and functionality of this component can modified as desired.

Properties

This component has standard Ignition properties with the addition of the following properties:

Data Properties

SPC Results Bind this property to the SPC Results property of the SPC Controller or SPC Selector components. Ther SPC Results contain all the information so that the control charts can display the results which is not possible with the data set alone.

```
Scripting name        spcResults
Data Type             SPCResults
```

SPC Data This property holds the SPC data and includes columns for the calculated values based on the SPC Data Format and selected attribute, control limits and signals.

```
Scripting name        spcData
Data Type             Dataset
```

Background Color The background color.

```
Scripting name        backgroundColor
Data Type             Color
```

No Data Message Text to display if not data is available to show in the control chart.

```
Scripting name        noDataMessage
Data Type             String
```

No Data Foreground The foreground color to display the no data message.

```
Scripting name        noDataForeground
Data Type             Color
```

No Data Font The font to display the no data message.

```
Scripting name        noDataFont
Data Type             Font
```

Chart Properties

Vertical If true, the bars will be shown vertically.

```
Scripting name        vertical
Data Type             boolean
```

© Inductive Automation

Chart Background Color The background color of the chart.

```
Scripting name        chartBackgroundColor
Data Type             Color
```

Bar Color Color of the bars.

```
Scripting name        barColor
Data Type             Color
```

Bar Spacing Specifies the spacing between the bars. It is a fractional value from 0.0 to 1.0 and represents the percentage of the bar width to make as space between the bars.

```
Scripting name        barSpacing
Data Type             float
```

Gradient If true, show bars with gradient fill.

```
Scripting name        gradient
Data Type             boolean
```

Shadows If true, show bars shadows for each bar.

```
Scripting name        shadows
Data Type             boolean
```

Tick Label Font The font to show values on the value and count axis.

```
Scripting name        tickLabelFont
Data Type             Font
```

Tick Label Color The color to show values on the value and count axis.

```
Scripting name        tickLabelColor
Data Type             Color
```

Value Axis Title The title to display on the value axis.

```
Scripting name        valueAxisTitle
Data Type             String
```

Frequency Axis Title The title to display on the frequency axis.

```
Scripting name        frequencyAxisTitle
Data Type             String
```

© Inductive Automation

Axis Title Font The font to show the axis titles.

```
Scripting name        axisTitleFont
Data Type             Font
```

Axis Title Color The color to show the axis titles.

```
Scripting name        axisTitleColor
Data Type             Color
```

Vertical Grid Line Color The color of the chart vertical grid lines.

```
Scripting name        verticalGridLineColor
Data Type             Color
```

Show Vertical Grid Lines If true, show the vertical grid lines in the charts.

```
Scripting name        showVerticalGridLines
Data Type             boolean
```

Horizontal Grid Line Color The color of the chart horizontal grid lines.

```
Scripting name        horizontalGridLineColor
Data Type             Color
```

Show Horizontal Grid Lines If true, show the horizontal grid lines in the charts.

```
Scripting name        showHorizontalGridLines
Data Type             boolean
```

Events

This component has standard Ignition events.

(none)

Methods

showSetLimitPanel()

Causes the calculate and set control limit dialog to be shown.

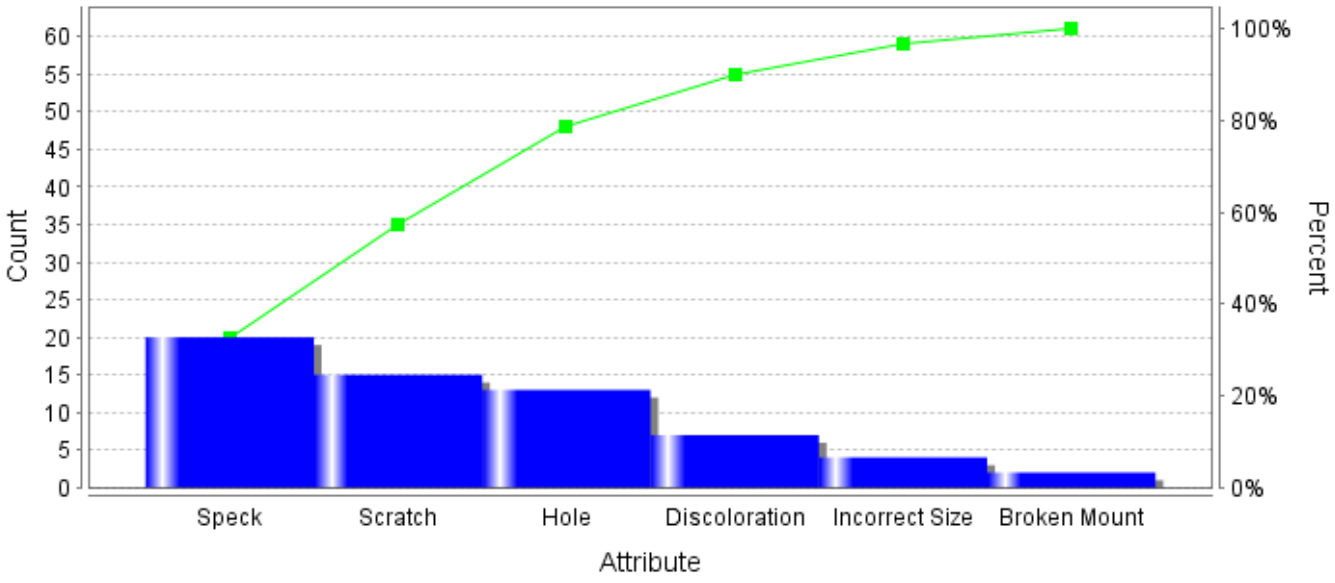
parameters (None)

returns nothing

3.4.2.5 Pareto Chart

Description

The Pareto chart is used to display which nonconforming items or nonconformities are the largest issue. It does not retrieve SPC results from the SPC module so it must be used with either the SPC Selector or the SPC Controller components that do. Only SPC results with Pareto SPC Data Format specified will be displayed.



Pareto Chart

Through the use of the properties listed below, the appearance and functionality of this component can modified as desired.

Properties

This component has standard Ignition properties with the addition of the following properties:

Data Properties

SPC Results

Bind this property to the SPC Results property of the SPC Controller or SPC Selector components. Ther SPC Results contain all the information so that the control charts can display the results which is not possible with the data set alone.

```
Scripting name      spcResults
Data Type           SPCResults
```

SPC Data

This property holds the SPC data and includes columns for the calculated values based on the SPC Data Format and selected attribute, control limits and signals.

```
Scripting name      spcData
Data Type           Dataset
```

© Inductive Automation

Background Color The background color.

```
Scripting name        backgroundColor
Data Type             Color
```

No Data Message Text to display if not data is available to show in the control chart.

```
Scripting name        noDataMessage
Data Type             String
```

No Data Foreground The foreground color to display the no data message.

```
Scripting name        noDataForeground
Data Type             Color
```

No Data Font The font to display the no data message.

```
Scripting name        noDataFont
Data Type             Font
```

Chart Properties

Vertical If true, the bars will be shown vertically.

```
Scripting name        vertical
Data Type             boolean
```

Chart Background Color The background color of the chart.

```
Scripting name        chartBackgroundColor
Data Type             Color
```

Bar Color Color of the bars.

```
Scripting name        barColor
Data Type             Color
```

Accumulation Line Color Color of the total accumulation line.

```
Scripting name        accumulationLineColor
Data Type             Color
```

Bar Spacing Specifies the spacing between the bars. It is a fractional value from 0.0 to 1.0 and represents the percentage of the bar width to make as space between the bars.

```
Scripting name        barSpacing
Data Type             float
```

© Inductive Automation

Gradient If true, show bars with gradient fill.

```
Scripting name        gradient
Data Type             boolean
```

Shadows If true, show bars shadows for each bar.

```
Scripting name        shadows
Data Type             boolean
```

Tick Label Font The font to show values on the value and count axis.

```
Scripting name        tickLabelFont
Data Type             Font
```

Tick Label Color The color to show values on the value and count axis.

```
Scripting name        tickLabelColor
Data Type             Color
```

Category Axis Title The title to display on the category axis.

```
Scripting name        categoryAxisTitle
Data Type             String
```

Frequency Axis Title The title to display on the frequency axis.

```
Scripting name        frequencyAxisTitle
Data Type             String
```

Axis Title Font The font to show the axis titles.

```
Scripting name        axisTitleFont
Data Type             Font
```

Axis Title Color The color to show the axis titles.

```
Scripting name        axisTitleColor
Data Type             Color
```

Horizontal Grid Line Color The color of the chart horizontal grid lines.

```
Scripting name        horizontalGridLineColor
Data Type             Color
```

Show Horizontal Grid Lines If true, show the horizontal grid lines in the charts.

```
Scripting name        showHorizontalGridLines
Data Type             boolean
```

© Inductive Automation

Show Accumulation Line If true, show the accumulation line in the chart.

```
Scripting name          showAccumulationLine
Data Type               boolean
```

Events
This component has standard Ignition events.

```
(none)
```

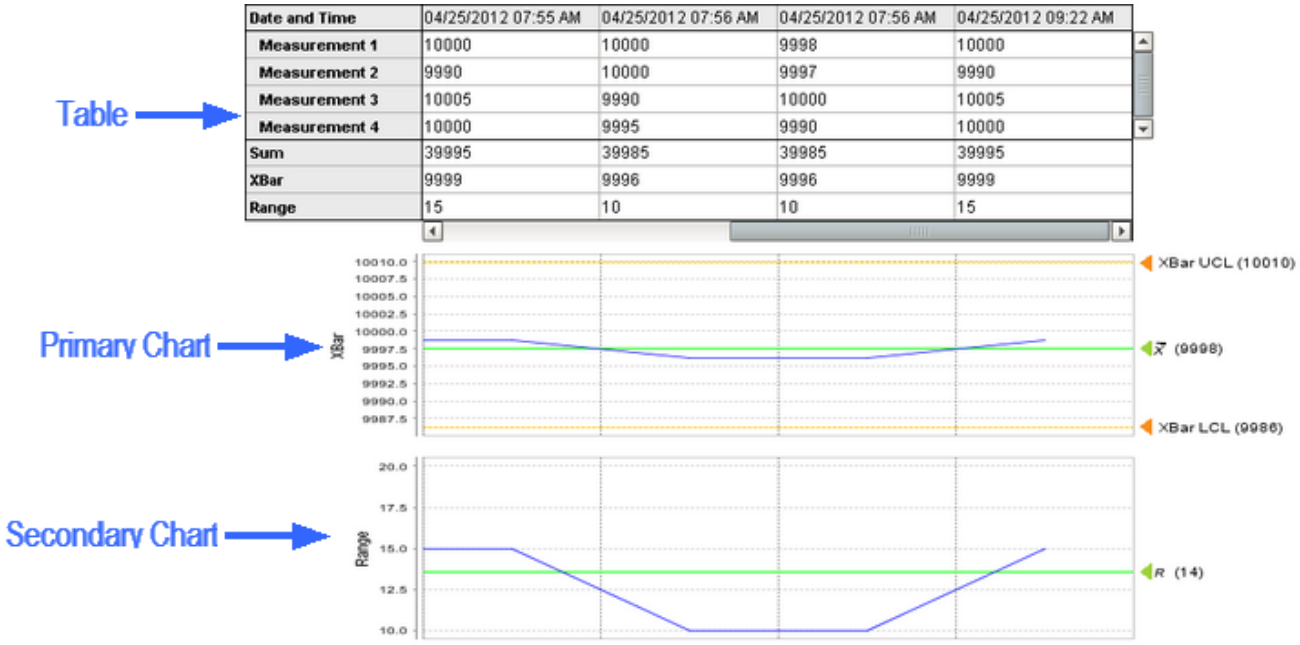
Methods

```
(none)
```

3.4.2.6 Xbar and R Chart

Description

The XBar Range control chart is used to display SPC results that have multiple measurements for each sample. It does not retrieve SPC results from the SPC module so it must be used with either the SPC Selector or the SPC Controller components that do. Only SPC results with XBar and Range SPC Data Format specified will be displayed.



XBar Range Control Chart

Through the use of the properties listed below, the appearance and functionality of this component can modified as desired.

Properties

This component has standard Ignition properties with the addition of the following properties:

Data Properties

SPC Results

Bind this property to the SPC Results property of the SPC Controller or SPC Selector components. Ther SPC Results contain all the information so that the control charts can display the results which is not possible with the data set alone.

```
Scripting name        spcResults
Data Type             SPCResults
```

SPC Data

This property holds the SPC data and includes columns for the calculated values based on the SPC Data Format and selected attribute, control limits and signals.

```
Scripting name        spcData
Data Type             Dataset
```

Measurement Count

This property represents the number of measurements for each sample in the SPC results.

```
Scripting name        measurementCount
Data Type             int
```

User

This property can optionally be set to override the current user logged in. It is used when notes or assignable causes are added or modified.

```
Scripting name        user
Data Type             String
```

Background Color

The background color.

```
Scripting name        backgroundColor
Data Type             Color
```

Definition Name

The sample definition to used when building SPC results.

```
Scripting name        definitionName
Data Type             String
```

No Data Message

Text to display if not data is available to show in the control chart.

```
Scripting name        noDataMessage
Data Type             String
```

© Inductive Automation

No Data Foreground The foreground color to display the no data message.

```
Scripting name        noDataForeground
Data Type             Color
```

No Data Font The font to display the no data message.

```
Scripting name        noDataFont
Data Type             Font
```

Table Properties

Show Table If true, the table containing measurement and calculated values will be shown at the top of the control chart.

```
Scripting name        showTable
Data Type             boolean
```

Min Visible Samples The minimum number of sample to show on the control chart. If more than the minimum visible samples exist in the SPC results, then a horizontal scroll bar will appear and allow the user to scroll back to earlier samples.

```
Scripting name        minVisibleSamples
Data Type             int
```

Min Visible Measurements The minimum number of measurements to show in the table of the control chart. If more than the minimum visible measurements exist in the SPC results, then a vertical scroll bar will appear and allow the user to through all measurements.

```
Scripting name        minVisibleMeasurements
Data Type             int
```

Column Width The width of the table column for each sample. The charts will be expanded to match the column width.

```
Scripting name        columnWidth
Data Type             int
```

Row Height The height of the table rows.

```
Scripting name        rowHeight
Data Type             int
```

Date Background The background color of the sample date row.

```
Scripting name        dateBackground
Data Type             Color
```

© Inductive Automation

Date Foreground The foreground color of the sample date values.

```
Scripting name        dateForeground
Data Type             Color
```

Date Font The font to display the sample date values.

```
Scripting name        dateFont
Data Type             Font
```

Date Format The date formatting pattern to display the sample dates.

```
Scripting name        dateFormat
Data Type             String
```

Label Background The background color of the labels.

```
Scripting name        labelBackground
Data Type             Color
```

Label Foreground The foreground color of the labels.

```
Scripting name        labelForeground
Data Type             Color
```

Label Font The font to display the labels.

```
Scripting name        labelFont
Data Type             Font
```

Data Background The background color of the measurement data values.

```
Scripting name        dataBackground
Data Type             Color
```

Data Foreground The foreground color of the measurement data values.

```
Scripting name        dataForeground
Data Type             Color
```

Data Font The font to display the measurement values.

```
Scripting name        dataFont
Data Type             Font
```

Calc Background The background color of the calculated data values.

```
Scripting name        calcBackground
Data Type             Color
```

© Inductive Automation

Calc Foreground The foreground color of the calculated data values.

```
Scripting name        calcForeground
Data Type             Color
```

Calc Font The font to display the calculated values.

```
Scripting name        calcFont
Data Type             Font
```

Chart Properties

Show Primary Chart If true, the primary chart will appear.

```
Scripting name        showPrimaryChart
Data Type             boolean
```

Show Secondary Chart If true, the secondary chart will appear.

```
Scripting name        showSecondaryChart
Data Type             boolean
```

Right Axis Width The width of the right chart axis in pixels.

```
Scripting name        rightAxisWidth
Data Type             int
```

Limit Dialog Horizontal Offset The horizontal, or x, position to display the set control limit dialog box.

```
Scripting name        limitDialogHorizontalOffset
Data Type             int
```

Limit Dialog Vertical Offset The vertical, or y, position to display the set control limit dialog box.

```
Scripting name        limitDialogVerticalOffset
Data Type             int
```

Vertical Grid Line Color The color of the chart vertical grid lines.

```
Scripting name        verticalGridLineColor
Data Type             Color
```

Show Vertical Grid Lines If true, show the vertical grid lines in the charts.

```
Scripting name        showVerticalGridLines
Data Type             boolean
```

© Inductive Automation

Horizontal Grid Line Color
The color of the chart horizontal grid lines.

```
Scripting name        horizontalGridLineColor
Data Type             Color
```

Show Horizontal Grid Lines
If true, show the horizontal grid lines in the charts.

```
Scripting name        showHorizontalGridLines
Data Type             boolean
```

Primary Chart Background
The background color of the primary chart.

```
Scripting name        primaryChartBackground
Data Type             Color
```

Secondary Chart Background
The background color of the secondary chart.

```
Scripting name        secondaryChartBackground
Data Type             Color
```

Show Notes
If true, show the note icon next to the chart point for any samples that have notes or assignable causes.

```
Scripting name        showNotes
Data Type             boolean
```

Note Image
The image to display next to the chart point for any samples that have notes or assignable causes.

```
Scripting name        noteImage
Data Type             Image
```

Enable Note Editing
If true, allow the user to add and edit notes and assignable causes.

```
Scripting name        enableNoteEditing
Data Type             boolean
```

Enable Point Deletion
If true, allow the user to temporarily remove samples from chart. This is used to remove samples that are known to out of control before calculating control limits. The sample that have been removed are not removed from the database and can be restored by selecting the Restore Points menu item.

```
Scripting name        enablePointDeletion
Data Type             boolean
```

Enable Control Limit Editing
If true, allow the user to calculate and set new control limit values.

```
Scripting name        enableControlLimitEditing
Data Type             boolean
```

Events

© Inductive Automation

This component has standard Ignition events.

`(none)`

Methods

`showSetLimitPanel(chartName)`

Causes the calculate and set control limit dialog to be shown.

parameters

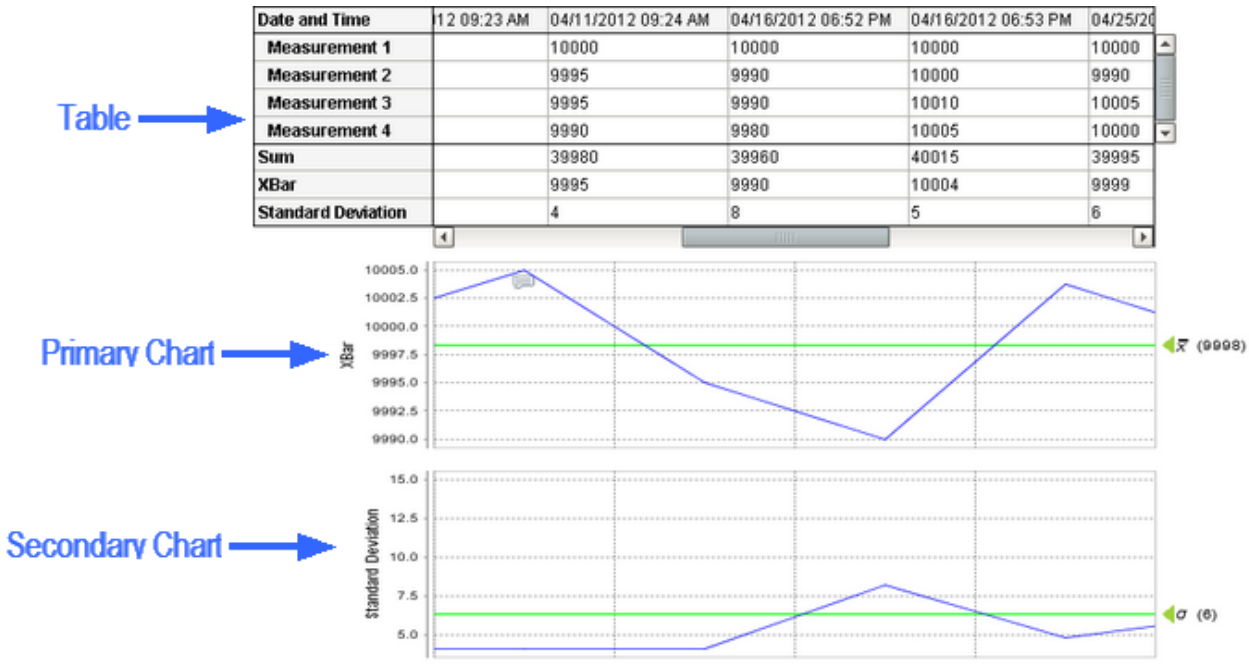
chartName	Which chart to show the control limit dialog for. Available c "Primary" or "Secondary"
	Data Type String

returns nothing

3.4.2.7 Xbar and S Chart

Description

The XBar Standard Deviation (S) control chart is used to display SPC results that have multiple measurements for each sample. It does not retrieve SPC results from the SPC module so it must be used with either the SPC Selector or the SPC Controller components that do. Only SPC results with XBar and S SPC Data Format specified will be displayed.



XBar Standard Deviation Control Chart

Through the use of the properties listed below, the appearance and functionality of this component can modified as desired.

Properties

This component has standard Ignition properties with the addition of the following properties:

Data Properties

SPC Results

Bind this property to the SPC Results property of the SPC Controller or SPC Selector components. Ther SPC Results contain all the information so that the control charts can display the results which is not possible with the data set alone.

```
Scripting name        spcResults
Data Type             SPCResults
```

SPC Data

This property holds the SPC data and includes columns for the calculated values based on the SPC Data Format and selected attribute, control limits and signals.

```
Scripting name        spcData
Data Type             Dataset
```

Measurement Count

This property represents the number of measurements for each sample in the SPC results.

```
Scripting name        measurementCount
Data Type             int
```

User

This property can optionally be set to override the current user logged in. It is used when notes or assignable causes are added or modified.

```
Scripting name        user
Data Type             String
```

Background Color

The background color.

```
Scripting name        backgroundColor
Data Type             Color
```

Definition Name

The sample definition to used when building SPC results.

```
Scripting name        definitionName
Data Type             String
```

No Data Message

Text to display if not data is available to show in the control chart.

```
Scripting name        noDataMessage
Data Type             String
```

© Inductive Automation

No Data Foreground The foreground color to display the no data message.

```
Scripting name        noDataForeground
Data Type             Color
```

No Data Font The font to display the no data message.

```
Scripting name        noDataFont
Data Type             Font
```

Table Properties

Show Table If true, the table containing measurement and calculated values will be shown at the top of the control chart.

```
Scripting name        showTable
Data Type             boolean
```

Min Visible Samples The minimum number of sample to show on the control chart. If more than the minimum visible samples exist in the SPC results, then a horizontal scroll bar will appear and allow the user to scroll back to earlier samples.

```
Scripting name        minVisibleSamples
Data Type             int
```

Min Visible Measurements The minimum number of measurements to show in the table of the control chart. If more than the minimum visible measurements exist in the SPC results, then a vertical scroll bar will appear and allow the user to through all measurements.

```
Scripting name        minVisibleMeasurements
Data Type             int
```

Column Width The width of the table column for each sample. The charts will be expanded to match the column width.

```
Scripting name        columnWidth
Data Type             int
```

Row Height The height of the table rows.

```
Scripting name        rowHeight
Data Type             int
```

Date Background The background color of the sample date row.

```
Scripting name        dateBackground
Data Type             Color
```

Date Foreground The foreground color of the sample date values.

```
Scripting name        dateForeground
Data Type             Color
```

© Inductive Automation

Date Font The font to display the sample date values.

```
Scripting name        dateFont
Data Type             Font
```

Date Format The date formatting pattern to display the sample dates.

```
Scripting name        dateFormat
Data Type             String
```

Label Background The background color of the labels.

```
Scripting name        labelBackground
Data Type             Color
```

Label Foreground The foreground color of the labels.

```
Scripting name        labelForeground
Data Type             Color
```

Label Font The font to display the labels.

```
Scripting name        labelFont
Data Type             Font
```

Data Background The background color of the measurement data values.

```
Scripting name        dataBackground
Data Type             Color
```

Data Foreground The foreground color of the measurement data values.

```
Scripting name        dataForeground
Data Type             Color
```

Data Font The font to display the measurement values.

```
Scripting name        dataFont
Data Type             Font
```

Calc Background The background color of the calculated data values.

```
Scripting name        calcBackground
Data Type             Color
```

© Inductive Automation

Calc Foreground The foreground color of the calculated data values.

```
Scripting name        calcForeground
Data Type             Color
```

Calc Font The font to display the calculated values.

```
Scripting name        calcFont
Data Type             Font
```

Chart Properties

Show Primary Chart If true, the primary chart will appear.

```
Scripting name        showPrimaryChart
Data Type             boolean
```

Show Secondary Chart If true, the secondary chart will appear.

```
Scripting name        showSecondaryChart
Data Type             boolean
```

Right Axis Width The width of the right chart axis in pixels.

```
Scripting name        rightAxisWidth
Data Type             int
```

Limit Dialog Horizontal Offset The horizontal, or x, position to display the set control limit dialog box.

```
Scripting name        limitDialogHorizontalOffset
Data Type             int
```

Limit Dialog Vertical Offset The vertical, or y, position to display the set control limit dialog box.

```
Scripting name        limitDialogVerticalOffset
Data Type             int
```

Vertical Grid Line Color The color of the chart vertical grid lines.

```
Scripting name        verticalGridLineColor
Data Type             Color
```

Show Vertical Grid Lines If true, show the vertical grid lines in the charts.

```
Scripting name        showVerticalGridLines
Data Type             boolean
```

© Inductive Automation

Horizontal Grid Line Color
The color of the chart horizontal grid lines.

```
Scripting name        horizontalGridLineColor
Data Type             Color
```

Show Horizontal Grid Lines
If true, show the horizontal grid lines in the charts.

```
Scripting name        showHorizontalGridLines
Data Type             boolean
```

Primary Chart Background
The background color of the primary chart.

```
Scripting name        primaryChartBackground
Data Type             Color
```

Secondary Chart Background
The background color of the secondary chart.

```
Scripting name        secondaryChartBackground
Data Type             Color
```

Show Notes
If true, show the note icon next to the chart point for any samples that have notes or assignable causes.

```
Scripting name        showNotes
Data Type             boolean
```

Note Image
The image to display next to the chart point for any samples that have notes or assignable causes.

```
Scripting name        noteImage
Data Type             Image
```

Enable Note Editing
If true, allow the user to add and edit notes and assignable causes.

```
Scripting name        enableNoteEditing
Data Type             boolean
```

Enable Point Deletion
If true, allow the user to temporarily remove samples from chart. This is used to remove samples that are known to out of control before calculating control limits. The sample that have been removed are not removed from the database and can be restored by selecting the Restore Points menu item.

```
Scripting name        enablePointDeletion
Data Type             boolean
```

Enable Control Limit Editing
If true, allow the user to calculate and set new control limit values.

```
Scripting name        enableControlLimitEditing
Data Type             boolean
```

Events

© Inductive Automation

This component has standard Ignition events.

`(none)`

Methods

`showSetLimitPanel(chartName)`

Causes the calculate and set control limit dialog to be shown.

parameters

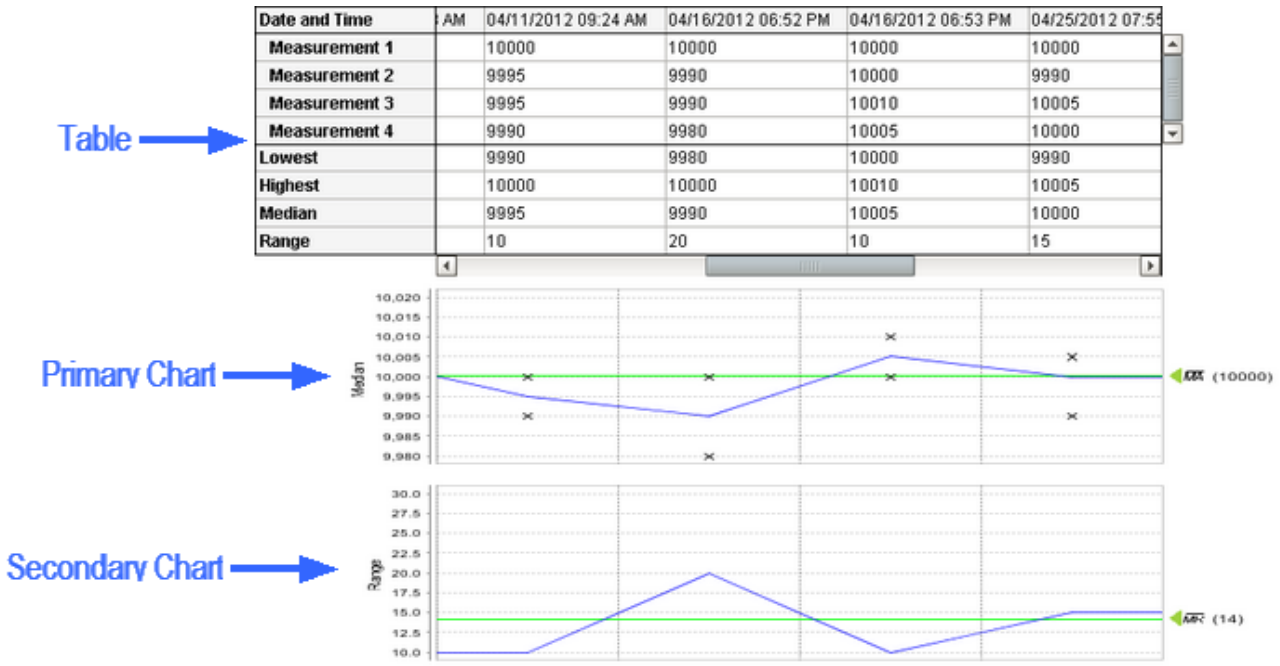
chartName	Which chart to show the control limit dialog for. Available c "Primary" or "Secondary"
	`Data Type String`

returns nothing

3.4.2.8 Median and Range Chart

Description

The Median Moving Range (MR) control chart is used to display SPC results that have multiple measurements for each sample. It does not retrieve SPC results from the SPC module so it must be used with either the SPC Selector or the SPC Controller components that do. Only SPC results with Median and MR SPC Data Format specified will be displayed.



Date and Time	AM	04/11/2012 09:24 AM	04/16/2012 06:52 PM	04/16/2012 06:53 PM	04/25/2012 07:55
Measurement 1		10000	10000	10000	10000
Measurement 2		9995	9990	10000	9990
Measurement 3		9995	9990	10010	10005
Measurement 4		9990	9980	10005	10000
Lowest		9990	9980	10000	9990
Highest		10000	10000	10010	10005
Median		9995	9990	10005	10000
Range		10	20	10	15

Median Moving Range Control Chart

Through the use of the properties listed below, the appearance and functionality of this component can modified as desired.

Properties

This component has standard Ignition properties with the addition of the following properties:

Data Properties

SPC Results

Bind this property to the SPC Results property of the SPC Controller or SPC Selector components. Ther SPC Results contain all the information so that the control charts can display the results which is not possible with the data set alone.

```
Scripting name        spcResults
Data Type             SPCResults
```

SPC Data

This property holds the SPC data and includes columns for the calculated values based on the SPC Data Format and selected attribute, control limits and signals.

```
Scripting name        spcData
Data Type             Dataset
```

Measurement Count

This property represents the number of measurements for each sample in the SPC results.

```
Scripting name        measurementCount
Data Type             int
```

User

This property can optionally be set to override the current user logged in. It is used when notes or assignable causes are added or modified.

```
Scripting name        user
Data Type             String
```

Background Color

The background color.

```
Scripting name        backgroundColor
Data Type             Color
```

Definition Name

The sample definition to used when building SPC results.

```
Scripting name        definitionName
Data Type             String
```

No Data Message

Text to display if not data is available to show in the control chart.

```
Scripting name        noDataMessage
Data Type             String
```

© Inductive Automation

No Data Foreground The foreground color to display the no data message.

```
Scripting name        noDataForeground
Data Type             Color
```

No Data Font The font to display the no data message.

```
Scripting name        noDataFont
Data Type             Font
```

Table Properties

Show Table If true, the table containing measurement and calculated values will be shown at the top of the control chart.

```
Scripting name        showTable
Data Type             boolean
```

Min Visible Samples The minimum number of sample to show on the control chart. If more than the minimum visible samples exist in the SPC results, then a horizontal scroll bar will appear and allow the user to scroll back to earlier samples.

```
Scripting name        minVisibleSamples
Data Type             int
```

Min Visible Measurements The minimum number of measurements to show in the table of the control chart. If more than the minimum visible measurements exist in the SPC results, then a vertical scroll bar will appear and allow the user to through all measurements.

```
Scripting name        minVisibleMeasurements
Data Type             int
```

Column Width The width of the table column for each sample. The charts will be expanded to match the column width.

```
Scripting name        columnWidth
Data Type             int
```

Row Height The height of the table rows.

```
Scripting name        rowHeight
Data Type             int
```

Date Background The background color of the sample date row.

```
Scripting name        dateBackground
Data Type             Color
```

© Inductive Automation

Date Foreground The foreground color of the sample date values.

```
Scripting name        dateForeground
Data Type             Color
```

Date Font The font to display the sample date values.

```
Scripting name        dateFont
Data Type             Font
```

Date Format The date formatting pattern to display the sample dates.

```
Scripting name        dateFormat
Data Type             String
```

Label Background The background color of the labels.

```
Scripting name        labelBackground
Data Type             Color
```

Label Foreground The foreground color of the labels.

```
Scripting name        labelForeground
Data Type             Color
```

Label Font The font to display the labels.

```
Scripting name        labelFont
Data Type             Font
```

Data Background The background color of the measurement data values.

```
Scripting name        dataBackground
Data Type             Color
```

Data Foreground The foreground color of the measurement data values.

```
Scripting name        dataForeground
Data Type             Color
```

Data Font The font to display the measurement values.

```
Scripting name        dataFont
Data Type             Font
```

Calc Background The background color of the calculated data values.

```
Scripting name        calcBackground
Data Type             Color
```

© Inductive Automation

Calc Foreground The foreground color of the calculated data values.

```
Scripting name        calcForeground
Data Type             Color
```

Calc Font The font to display the calculated values.

```
Scripting name        calcFont
Data Type             Font
```

Chart Properties

Show Primary Chart If true, the primary chart will appear.

```
Scripting name        showPrimaryChart
Data Type             boolean
```

Show Secondary Chart If true, the secondary chart will appear.

```
Scripting name        showSecondaryChart
Data Type             boolean
```

Right Axis Width The width of the right chart axis in pixels.

```
Scripting name        rightAxisWidth
Data Type             int
```

Limit Dialog Horizontal Offset The horizontal, or x, position to display the set control limit dialog box.

```
Scripting name        limitDialogHorizontalOffset
Data Type             int
```

Limit Dialog Vertical Offset The vertical, or y, position to display the set control limit dialog box.

```
Scripting name        limitDialogVerticalOffset
Data Type             int
```

Vertical Grid Line Color The color of the chart vertical grid lines.

```
Scripting name        verticalGridLineColor
Data Type             Color
```

Show Vertical Grid Lines If true, show the vertical grid lines in the charts.

```
Scripting name        showVerticalGridLines
Data Type             boolean
```

© Inductive Automation

Horizontal Grid Line Color The color of the chart horizontal grid lines.

```
Scripting name        horizontalGridLineColor
Data Type             Color
```

Show Horizontal Grid Lines If true, show the horizontal grid lines in the charts.

```
Scripting name        showHorizontalGridLines
Data Type             boolean
```

Primary Chart Background The background color of the primary chart.

```
Scripting name        primaryChartBackground
Data Type             Color
```

Secondary Chart Background The background color of the secondary chart.

```
Scripting name        secondaryChartBackground
Data Type             Color
```

Show Notes If true, show the note icon next to the chart point for any samples that have notes or assignable causes.

```
Scripting name        showNotes
Data Type             boolean
```

Note Image The image to display next to the chart point for any samples that have notes or assignable causes.

```
Scripting name        noteImage
Data Type             Image
```

Enable Note Editing If true, allow the user to add and edit notes and assignable causes.

```
Scripting name        enableNoteEditing
Data Type             boolean
```

Enable Point Deletion If true, allow the user to temporarily remove samples from chart. This is used to remove samples that are known to out of control before calculating control limits. The sample that have been removed are not removed from the database and can be restored by selecting the Restore Points menu item.

```
Scripting name        enablePointDeletion
Data Type             boolean
```

Enable Control Limit Editing If true, allow the user to calculate and set new control limit values.

```
Scripting name        enableControlLimitEditing
Data Type             boolean
```

Events

© Inductive Automation

This component has standard Ignition events.

`(none)`

Methods

`showSetLimitPanel(chartName)`

Causes the calculate and set control limit dialog to be shown.

parameters

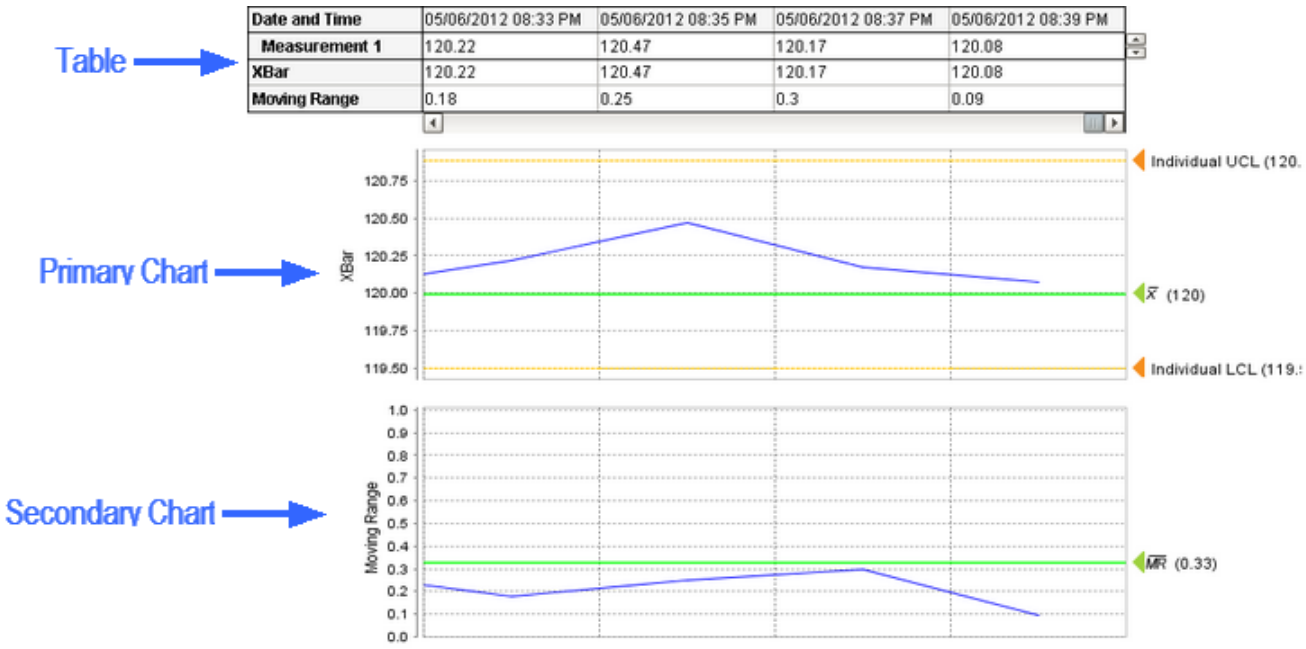
chartName	Which chart to show the control limit dialog for. Available c "Primary" or "Secondary"
	Data Type String

returns nothing

3.4.2.9 Individual and Range Chart

Description

The Individual Moving Range (MR) control chart is used to display SPC results that have a single measurement for each sample. It does not retrieve SPC results from the SPC module so it must be used with either the SPC Selector or the SPC Controller components that do. Only SPC results with Individual and MR SPC Data Format specified will be displayed.



Individual Moving Range Control Chart

Through the use of the properties listed below, the appearance and functionality of this component can modified as desired.

Properties

This component has standard Ignition properties with the addition of the following properties:

Data Properties

SPC Results

Bind this property to the SPC Results property of the SPC Controller or SPC Selector components. Ther SPC Results contain all the information so that the control charts can display the results which is not possible with the data set alone.

```
Scripting name        spcResults
Data Type             SPCResults
```

SPC Data

This property holds the SPC data and includes columns for the calculated values based on the SPC Data Format and selected attribute, control limits and signals.

```
Scripting name        spcData
Data Type             Dataset
```

Measurement Count

This property represents the number of measurements for each sample in the SPC results.

```
Scripting name        measurementCount
Data Type             int
```

User

This property can optionally be set to override the current user logged in. It is used when notes or assignable causes are added or modified.

```
Scripting name        user
Data Type             String
```

Background Color

The background color.

```
Scripting name        backgroundColor
Data Type             Color
```

Definition Name

The sample definition to used when building SPC results.

```
Scripting name        definitionName
Data Type             String
```

No Data Message

Text to display if not data is available to show in the control chart.

```
Scripting name        noDataMessage
Data Type             String
```

© Inductive Automation

No Data Foreground The foreground color to display the no data message.

```
Scripting name        noDataForeground
Data Type             Color
```

No Data Font The font to display the no data message.

```
Scripting name        noDataFont
Data Type             Font
```

Table Properties

Show Table If true, the table containing measurement and calculated values will be shown at the top of the control chart.

```
Scripting name        showTable
Data Type             boolean
```

Min Visible Samples The minimum number of sample to show on the control chart. If more than the minimum visible samples exist in the SPC results, then a horizontal scroll bar will appear and allow the user to scroll back to earlier samples.

```
Scripting name        minVisibleSamples
Data Type             int
```

Min Visible Measurements The minimum number of measurements to show in the table of the control chart. If more than the minimum visible measurements exist in the SPC results, then a vertical scroll bar will appear and allow the user to through all measurements.

```
Scripting name        minVisibleMeasurements
Data Type             int
```

Column Width The width of the table column for each sample. The charts will be expanded to match the column width.

```
Scripting name        columnWidth
Data Type             int
```

Row Height The height of the table rows.

```
Scripting name        rowHeight
Data Type             int
```

Date Background The background color of the sample date row.

```
Scripting name        dateBackground
Data Type             Color
```

© Inductive Automation

Date Foreground The foreground color of the sample date values.

```
Scripting name        dateForeground
Data Type             Color
```

Date Font The font to display the sample date values.

```
Scripting name        dateFont
Data Type             Font
```

Date Format The date formatting pattern to display the sample dates.

```
Scripting name        dateFormat
Data Type             String
```

Label Background The background color of the labels.

```
Scripting name        labelBackground
Data Type             Color
```

Label Foreground The foreground color of the labels.

```
Scripting name        labelForeground
Data Type             Color
```

Label Font The font to display the labels.

```
Scripting name        labelFont
Data Type             Font
```

Data Background The background color of the measurement data values.

```
Scripting name        dataBackground
Data Type             Color
```

Data Foreground The foreground color of the measurement data values.

```
Scripting name        dataForeground
Data Type             Color
```

Data Font The font to display the measurement values.

```
Scripting name        dataFont
Data Type             Font
```

Calc Background The background color of the calculated data values.

```
Scripting name        calcBackground
Data Type             Color
```

© Inductive Automation

Calc Foreground The foreground color of the calculated data values.

```
Scripting name        calcForeground
Data Type             Color
```

Calc Font The font to display the calculated values.

```
Scripting name        calcFont
Data Type             Font
```

Chart Properties

Show Primary Chart If true, the primary chart will appear.

```
Scripting name        showPrimaryChart
Data Type             boolean
```

Show Secondary Chart If true, the secondary chart will appear.

```
Scripting name        showSecondaryChart
Data Type             boolean
```

Right Axis Width The width of the right chart axis in pixels.

```
Scripting name        rightAxisWidth
Data Type             int
```

Limit Dialog Horizontal Offset The horizontal, or x, position to display the set control limit dialog box.

```
Scripting name        limitDialogHorizontalOffset
Data Type             int
```

Limit Dialog Vertical Offset The vertical, or y, position to display the set control limit dialog box.

```
Scripting name        limitDialogVerticalOffset
Data Type             int
```

Vertical Grid Line Color The color of the chart vertical grid lines.

```
Scripting name        verticalGridLineColor
Data Type             Color
```

Show Vertical Grid Lines If true, show the vertical grid lines in the charts.

```
Scripting name        showVerticalGridLines
Data Type             boolean
```

© Inductive Automation

Horizontal Grid Line Color The color of the chart horizontal grid lines.

```
Scripting name        horizontalGridLineColor
Data Type             Color
```

Show Horizontal Grid Lines If true, show the horizontal grid lines in the charts.

```
Scripting name        showHorizontalGridLines
Data Type             boolean
```

Primary Chart Background The background color of the primary chart.

```
Scripting name        primaryChartBackground
Data Type             Color
```

Secondary Chart Background The background color of the secondary chart.

```
Scripting name        secondaryChartBackground
Data Type             Color
```

Show Notes If true, show the note icon next to the chart point for any samples that have notes or assignable causes.

```
Scripting name        showNotes
Data Type             boolean
```

Note Image The image to display next to the chart point for any samples that have notes or assignable causes.

```
Scripting name        noteImage
Data Type             Image
```

Enable Note Editing If true, allow the user to add and edit notes and assignable causes.

```
Scripting name        enableNoteEditing
Data Type             boolean
```

Enable Point Deletion If true, allow the user to temporarily remove samples from chart. This is used to remove samples that are known to out of control before calculating control limits. The sample that have been removed are not removed from the database and can be restored by selecting the Restore Points menu item.

```
Scripting name        enablePointDeletion
Data Type             boolean
```

Enable Control Limit Editing If true, allow the user to calculate and set new control limit values.

```
Scripting name        enableControlLimitEditing
Data Type             boolean
```

Events

© Inductive Automation

This component has standard Ignition events.

```
(none)
```

Methods

```
showSetLimitPanel(chartName)
```

Causes the calculate and set control limit dialog to be shown.

parameters

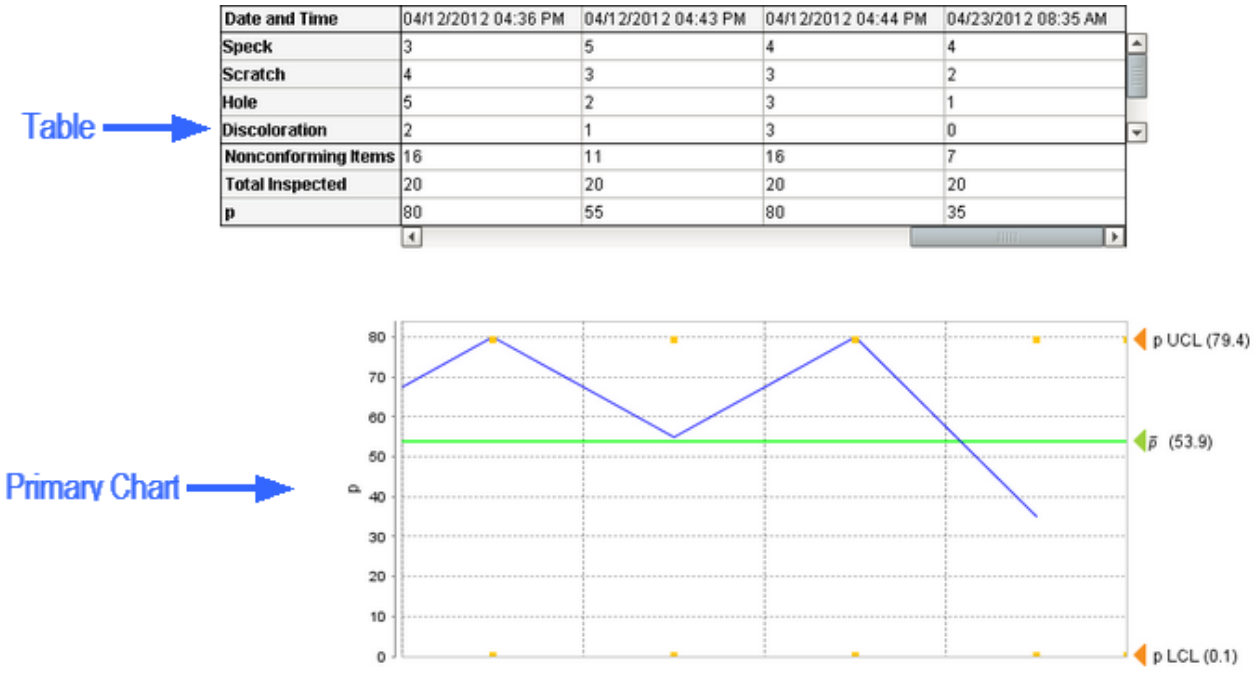
chartName	Which chart to show the control limit dialog for. Available c "Primary" or "Secondary"
	Data Type String

returns nothing

3.4.2.10 P-Chart

Description

The Percentage of Nonconforming Items (p) control chart is used to display SPC results that have nonconforming counts for each sample. It does not retrieve SPC results from the SPC module so it must be used with either the SPC Selector or the SPC Controller components that do. Only SPC results with p chart SPC Data Format specified will be displayed.



P Control Chart

Through the use of the properties listed below, the appearance and functionality of this component can modified as desired.

Properties

This component has standard Ignition properties with the addition of the following properties:

Data Properties

SPC Results

Bind this property to the SPC Results property of the SPC Controller or SPC Selector components. Ther SPC Results contain all the information so that the control charts can display the results which is not possible with the data set alone.

```
Scripting name        spcResults
Data Type             SPCResults
```

SPC Data

This property holds the SPC data and includes columns for the calculated values based on the SPC Data Format and selected attribute, control limits and signals.

```
Scripting name        spcData
Data Type             Dataset
```

Measurement Count

This property represents the number of measurements for each sample in the SPC results.

```
Scripting name        measurementCount
Data Type             int
```

User

This property can optionally be set to override the current user logged in. It is used when notes or assignable causes are added or modified.

```
Scripting name        user
Data Type             String
```

Background Color

The background color.

```
Scripting name        backgroundColor
Data Type             Color
```

Definition Name

The sample definition to used when building SPC results.

```
Scripting name        definitionName
Data Type             String
```

No Data Message

Text to display if not data is available to show in the control chart.

```
Scripting name        noDataMessage
Data Type             String
```

© Inductive Automation

No Data Foreground The foreground color to display the no data message.

```
Scripting name        noDataForeground
Data Type             Color
```

No Data Font The font to display the no data message.

```
Scripting name        noDataFont
Data Type             Font
```

Table Properties

Show Table If true, the table containing measurement and calculated values will be shown at the top of the control chart.

```
Scripting name        showTable
Data Type             boolean
```

Min Visible Samples The minimum number of sample to show on the control chart. If more than the minimum visible samples exist in the SPC results, then a horizontal scroll bar will appear and allow the user to scroll back to earlier samples.

```
Scripting name        minVisibleSamples
Data Type             int
```

Min Visible Measurements The minimum number of measurements to show in the table of the control chart. If more than the minimum visible measurements exist in the SPC results, then a vertical scroll bar will appear and allow the user to through all measurements.

```
Scripting name        minVisibleMeasurements
Data Type             int
```

Column Width The width of the table column for each sample. The charts will be expanded to match the column width.

```
Scripting name        columnWidth
Data Type             int
```

Row Height The height of the table rows.

```
Scripting name        rowHeight
Data Type             int
```

Date Background The background color of the sample date row.

```
Scripting name        dateBackground
Data Type             Color
```

© Inductive Automation

Date Foreground The foreground color of the sample date values.

```
Scripting name        dateForeground
Data Type             Color
```

Date Font The font to display the sample date values.

```
Scripting name        dateFont
Data Type             Font
```

Date Format The date formatting pattern to display the sample dates.

```
Scripting name        dateFormat
Data Type             String
```

Label Background The background color of the labels.

```
Scripting name        labelBackground
Data Type             Color
```

Label Foreground The foreground color of the labels.

```
Scripting name        labelForeground
Data Type             Color
```

Label Font The font to display the labels.

```
Scripting name        labelFont
Data Type             Font
```

Data Background The background color of the measurement data values.

```
Scripting name        dataBackground
Data Type             Color
```

Data Foreground The foreground color of the measurement data values.

```
Scripting name        dataForeground
Data Type             Color
```

Data Font The font to display the measurement values.

```
Scripting name        dataFont
Data Type             Font
```

Calc Background The background color of the calculated data values.

```
Scripting name        calcBackground
Data Type             Color
```

© Inductive Automation

Calc Foreground The foreground color of the calculated data values.

```
Scripting name        calcForeground
Data Type             Color
```

Calc Font The font to display the calculated values.

```
Scripting name        calcFont
Data Type             Font
```

Chart Properties

Show Primary Chart If true, the primary chart will appear.

```
Scripting name        showPrimaryChart
Data Type             boolean
```

Right Axis Width The width of the right chart axis in pixels.

```
Scripting name        rightAxisWidth
Data Type             int
```

Limit Dialog Horizontal Offset The horizontal, or x, position to display the set control limit dialog box.

```
Scripting name        limitDialogHorizontalOffset
Data Type             int
```

Limit Dialog Vertical Offset The vertical, or y, position to display the set control limit dialog box.

```
Scripting name        limitDialogVerticalOffset
Data Type             int
```

Vertical Grid Line Color The color of the chart vertical grid lines.

```
Scripting name        verticalGridLineColor
Data Type             Color
```

Show Vertical Grid Lines If true, show the vertical grid lines in the charts.

```
Scripting name        showVerticalGridLines
Data Type             boolean
```

Horizontal Grid Line Color The color of the chart horizontal grid lines.

```
Scripting name        horizontalGridLineColor
Data Type             Color
```

© Inductive Automation

Show Horizontal Grid Lines If true, show the horizontal grid lines in the charts.

```
Scripting name        showHorizontalGridLines
Data Type             boolean
```

Primary Chart Background The background color of the primary chart.

```
Scripting name        primaryChartBackground
Data Type             Color
```

Show Notes If true, show the note icon next to the chart point for any samples that have notes or assignable causes.

```
Scripting name        showNotes
Data Type             boolean
```

Note Image The image to display next to the chart point for any samples that have notes or assignable causes.

```
Scripting name        noteImage
Data Type             Image
```

Enable Note Editing If true, allow the user to add and edit notes and assignable causes.

```
Scripting name        enableNoteEditing
Data Type             boolean
```

Enable Point Deletion If true, allow the user to temporarily remove samples from chart. This is used to remove samples that are known to out of control before calculating control limits. The sample that have been removed are not removed from the database and can be restored by selecting the Restore Points menu item.

```
Scripting name        enablePointDeletion
Data Type             boolean
```

Enable Control Limit Editing If true, allow the user to calculate and set new control limit values.

```
Scripting name        enableControlLimitEditing
Data Type             boolean
```

Events

This component has standard Ignition events.

(none)

Methods

showSetLimitPanel(chartName)

Causes the calculate and set control limit dialog to be shown.

© Inductive Automation

parameters

chartName	Which chart to show the control limit dialog for. Available c "Primary".
	Data Type String

returns nothing

3.4.2.11 NP-Chart

Description

The Number of Nonconforming Items (np) control chart is used to display SPC results that have nonconforming counts for each sample. It does not retrieve SPC results from the SPC module so it must be used with either the SPC Selector or the SPC Controller components that do. Only SPC results with np chart SPC Data Format specified will be displayed.

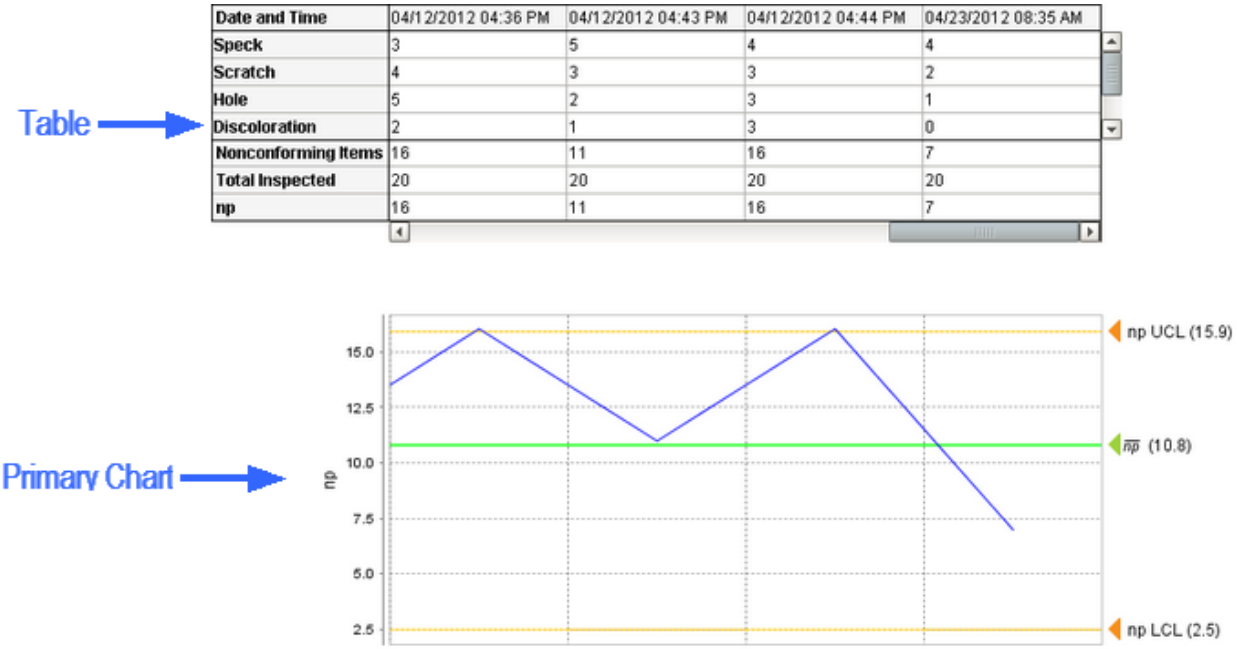


Table
Primary Chart
Date and Time | 04/12/2012 04:36 PM | 04/12/2012 04:43 PM | 04/12/2012 04:44 PM | 04/23/2012 08:35 AM
Speck | 3 | 5 | 4 | 4
Scratch | 4 | 3 | 3 | 2
Hole | 5 | 2 | 3 | 1
Discoloration | 2 | 1 | 3 | 0
Nonconforming Items | 16 | 11 | 16 | 7
Total Inspected | 20 | 20 | 20 | 20
np | 16 | 11 | 16 | 7
np
15.0
12.5
10.0
7.5
5.0
2.5
np UCL (15.9)
np (10.8)
np LCL (2.5)

NP Control Chart

Through the use of the properties listed below, the appearance and functionality of this component can modified as desired.

Properties

This component has standard Ignition properties with the addition of the following properties:

Data Properties

SPC Results Bind this property to the SPC Results property of the SPC Controller or SPC Selector components. Ther SPC Results contain all the information so that the control charts can display the results which is not possible with the data set alone.

```
Scripting name        spcResults
Data Type             SPCResults
```

SPC Data This property holds the SPC data and includes columns for the calculated values based on the SPC Data Format and selected attribute, control limits and signals.

```
Scripting name        spcData
Data Type             Dataset
```

Measurement Count This property represents the number of measurements for each sample in the SPC results.

```
Scripting name        measurementCount
Data Type             int
```

User This property can optionally be set to override the current user logged in. It is used when notes or assignable causes are added or modified.

```
Scripting name        user
Data Type             String
```

Background Color The background color.

```
Scripting name        backgroundColor
Data Type             Color
```

Definition Name The sample definition to used when building SPC results.

```
Scripting name        definitionName
Data Type             String
```

No Data Message Text to display if not data is available to show in the control chart.

```
Scripting name        noDataMessage
Data Type             String
```

No Data Foreground The foreground color to display the no data message.

```
Scripting name        noDataForeground
Data Type             Color
```

No Data Font The font to display the no data message.

```
Scripting name        noDataFont
Data Type             Font
```

Table Properties

© Inductive Automation

Show Table If true, the table containing measurement and calculated values will be shown at the top of the control chart.

```
Scripting name        showTable
Data Type             boolean
```

Min Visible Samples The minimum number of sample to show on the control chart. If more than the minimum visible samples exist in the SPC results, then a horizontal scroll bar will appear and allow the user to scroll back to earlier samples.

```
Scripting name        minVisibleSamples
Data Type             int
```

Min Visible Measurements The minimum number of measurements to show in the table of the control chart. If more than the minimum visible measurements exist in the SPC results, then a vertical scroll bar will appear and allow the user to through all measurements.

```
Scripting name        minVisibleMeasurements
Data Type             int
```

Column Width The width of the table column for each sample. The charts will be expanded to match the column width.

```
Scripting name        columnWidth
Data Type             int
```

Row Height The height of the table rows.

```
Scripting name        rowHeight
Data Type             int
```

Date Background The background color of the sample date row.

```
Scripting name        dateBackground
Data Type             Color
```

Date Foreground The foreground color of the sample date values.

```
Scripting name        dateForeground
Data Type             Color
```

Date Font The font to display the sample date values.

```
Scripting name        dateFont
Data Type             Font
```

Date Format The date formatting pattern to display the sample dates.

```
Scripting name        dateFormat
Data Type             String
```

© Inductive Automation

Label Background The background color of the labels.

```
Scripting name        labelBackground
Data Type             Color
```

Label Foreground The foreground color of the labels.

```
Scripting name        labelForeground
Data Type             Color
```

Label Font The font to display the labels.

```
Scripting name        labelFont
Data Type             Font
```

Data Background The background color of the measurement data values.

```
Scripting name        dataBackground
Data Type             Color
```

Data Foreground The foreground color of the measurement data values.

```
Scripting name        dataForeground
Data Type             Color
```

Data Font The font to display the measurement values.

```
Scripting name        dataFont
Data Type             Font
```

Calc Background The background color of the calculated data values.

```
Scripting name        calcBackground
Data Type             Color
```

Calc Foreground The foreground color of the calculated data values.

```
Scripting name        calcForeground
Data Type             Color
```

Calc Font The font to display the calculated values.

```
Scripting name        calcFont
Data Type             Font
```

Chart Properties

© Inductive Automation

Show Primary Chart

If true, the primary chart will appear.

```
Scripting name        showPrimaryChart
Data Type             boolean
```

Right Axis Width

The width of the right chart axis in pixels.

```
Scripting name        rightAxisWidth
Data Type             int
```

Limit Dialog Horizontal Offset

The horizontal, or x, position to display the set control limit dialog box.

```
Scripting name        limitDialogHorizontalOffset
Data Type             int
```

Limit Dialog Vertical Offset

The vertical, or y, position to display the set control limit dialog box.

```
Scripting name        limitDialogVerticalOffset
Data Type             int
```

Vertical Grid Line Color

The color of the chart vertical grid lines.

```
Scripting name        verticalGridLineColor
Data Type             Color
```

Show Vertical Grid Lines

If true, show the vertical grid lines in the charts.

```
Scripting name        showVerticalGridLines
Data Type             boolean
```

Horizontal Grid Line Color

The color of the chart horizontal grid lines.

```
Scripting name        horizontalGridLineColor
Data Type             Color
```

Show Horizontal Grid Lines

If true, show the horizontal grid lines in the charts.

```
Scripting name        showHorizontalGridLines
Data Type             boolean
```

Primary Chart Background

The background color of the primary chart.

```
Scripting name        primaryChartBackground
Data Type             Color
```

Show Notes

If true, show the note icon next to the chart point for any samples that have notes or assignable causes.

```
Scripting name        showNotes
Data Type             boolean
```

© Inductive Automation

Note Image
The image to display next to the chart point for any samples that have notes or assignable causes.

```
Scripting name        noteImage
Data Type             Image
```

Enable Note Editing
If true, allow the user to add and edit notes and assignable causes.

```
Scripting name        enableNoteEditing
Data Type             boolean
```

Enable Point Deletion
If true, allow the user to temporarily remove samples from chart. This is used to remove samples that are known to out of control before calculating control limits. The sample that have been removed are not removed from the database and can be restored by selecting the Restore Points menu item.

```
Scripting name        enablePointDeletion
Data Type             boolean
```

Enable Control Limit Editing
If true, allow the user to calculate and set new control limit values.

```
Scripting name        enableControlLimitEditing
Data Type             boolean
```

Events
This component has standard Ignition events.

(none)

Methods

showSetLimitPanel(chartName)

Causes the calculate and set control limit dialog to be shown.

parameters

chartName — Which chart to show the control limit dialog for. Available c "Primary".

```
Data Type             String
```

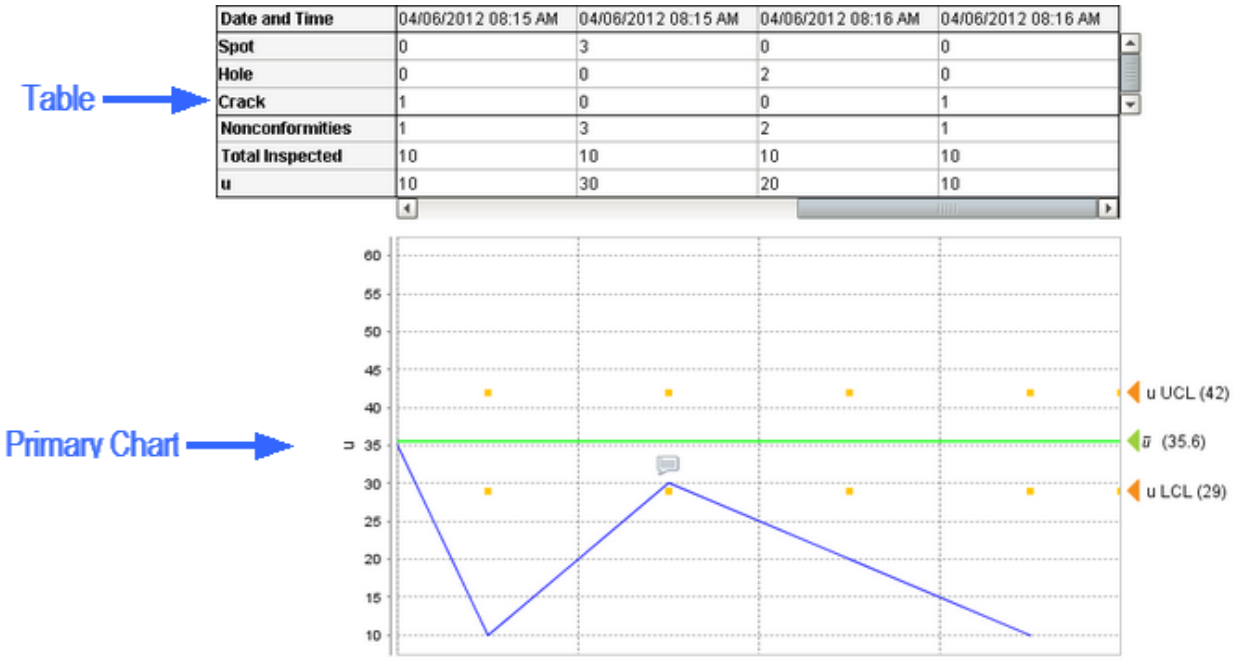
returns nothing

3.4.2.12 U-Chart

Description

The Percentage of Nonconformities (u) control chart is used to display SPC results that have nonconformities counts for each sample. It does not retrieve SPC results from the SPC module so it

must be used with either the SPC Selector or the SPC Controller components that do. Only SPC results with u chart SPC Data Format specified will be displayed.



Date and Time	04/06/2012 08:15 AM	04/06/2012 08:15 AM	04/06/2012 08:16 AM	04/06/2012 08:16 AM
Spot	0	3	0	0
Hole	0	0	2	0
Crack	1	0	0	1
Nonconformities	1	3	2	1
Total Inspected	10	10	10	10
u	10	30	20	10

U Control Chart

Through the use of the properties listed below, the appearance and functionality of this component can modified as desired.

Properties

This component has standard Ignition properties with the addition of the following properties:

Data Properties

SPC Results

Bind this property to the SPC Results property of the SPC Controller or SPC Selector components. Ther SPC Results contain all the information so that the control charts can display the results which is not possible with the data set alone.

```
Scripting name      spcResults
Data Type           SPCResults
```

SPC Data

This property holds the SPC data and includes columns for the calculated values based on the SPC Data Format and selected attribute, control limits and signals.

```
Scripting name      spcData
Data Type           Dataset
```

Measurement Count

This property represents the number of measurements for each sample in the SPC results.

```
Scripting name      measurementCount
Data Type           int
```

© Inductive Automation

User This property can optionally be set to override the current user logged in. It is used when notes or assignable causes are added or modified.

```
Scripting name        user
Data Type             String
```

Background Color The background color.

```
Scripting name        backgroundColor
Data Type             Color
```

Definition Name The sample definition to used when building SPC results.

```
Scripting name        definitionName
Data Type             String
```

No Data Message Text to display if not data is available to show in the control chart.

```
Scripting name        noDataMessage
Data Type             String
```

No Data Foreground The foreground color to display the no data message.

```
Scripting name        noDataForeground
Data Type             Color
```

No Data Font The font to display the no data message.

```
Scripting name        noDataFont
Data Type             Font
```

Table Properties

Show Table If true, the table containing measurement and calculated values will be shown at the top of the control chart.

```
Scripting name        showTable
Data Type             boolean
```

Min Visible Samples The minimum number of sample to show on the control chart. If more than the minimum visible samples exist in the SPC results, then a horizontal scroll bar will appear and allow the user to scroll back to earlier samples.

```
Scripting name        minVisibleSamples
Data Type             int
```

© Inductive Automation

Min Visible Measurements

The minimum number of measurements to show in the table of the control chart. If more than the minimum visible measurements exist in the SPC results, then a vertical scroll bar will appear and allow the user to through all measurements.

```
Scripting name        minVisibleMeasurements
Data Type             int
```

Column Width

The width of the table column for each sample. The charts will be expanded to match the column width.

```
Scripting name        columnWidth
Data Type             int
```

Row Height

The height of the table rows.

```
Scripting name        rowHeight
Data Type             int
```

Date Background

The background color of the sample date row.

```
Scripting name        dateBackground
Data Type             Color
```

Date Foreground

The foreground color of the sample date values.

```
Scripting name        dateForeground
Data Type             Color
```

Date Font

The font to display the sample date values.

```
Scripting name        dateFont
Data Type             Font
```

Date Format

The date formatting pattern to display the sample dates.

```
Scripting name        dateFormat
Data Type             String
```

Label Background

The background color of the labels.

```
Scripting name        labelBackground
Data Type             Color
```

Label Foreground

The foreground color of the labels.

```
Scripting name        labelForeground
Data Type             Color
```

© Inductive Automation

Label Font

The font to display the labels.

```
Scripting name        labelFont
Data Type             Font
```

Data Background

The background color of the measurement data values.

```
Scripting name        dataBackground
Data Type             Color
```

Data Foreground

The foreground color of the measurement data values.

```
Scripting name        dataForeground
Data Type             Color
```

Data Font

The font to display the measurement values.

```
Scripting name        dataFont
Data Type             Font
```

Calc Background

The background color of the calculated data values.

```
Scripting name        calcBackground
Data Type             Color
```

Calc Foreground

The foreground color of the calculated data values.

```
Scripting name        calcForeground
Data Type             Color
```

Calc Font

The font to display the calculated values.

```
Scripting name        calcFont
Data Type             Font
```

Chart Properties

Show Primary Chart

If true, the primary chart will appear.

```
Scripting name        showPrimaryChart
Data Type             boolean
```

Right Axis Width

The width of the right chart axis in pixels.

```
Scripting name        rightAxisWidth
Data Type             int
```

© Inductive Automation

Limit Dialog Horizontal Offset

The horizontal, or x, position to display the set control limit dialog box.

```
Scripting name        limitDialogHorizontalOffset
Data Type             int
```

Limit Dialog Vertical Offset

The vertical, or y, position to display the set control limit dialog box.

```
Scripting name        limitDialogVerticalOffset
Data Type             int
```

Vertical Grid Line Color

The color of the chart vertical grid lines.

```
Scripting name        verticalGridLineColor
Data Type             Color
```

Show Vertical Grid Lines

If true, show the vertical grid lines in the charts.

```
Scripting name        showVerticalGridLines
Data Type             boolean
```

Horizontal Grid Line Color

The color of the chart horizontal grid lines.

```
Scripting name        horizontalGridLineColor
Data Type             Color
```

Show Horizontal Grid Lines

If true, show the horizontal grid lines in the charts.

```
Scripting name        showHorizontalGridLines
Data Type             boolean
```

Primary Chart Background

The background color of the primary chart.

```
Scripting name        primaryChartBackground
Data Type             Color
```

Show Notes

If true, show the note icon next to the chart point for any samples that have notes or assignable causes.

```
Scripting name        showNotes
Data Type             boolean
```

Note Image

The image to display next to the chart point for any samples that have notes or assignable causes.

```
Scripting name        noteImage
Data Type             Image
```

Enable Note Editing

If true, allow the user to add and edit notes and assignable causes.

```
Scripting name        enableNoteEditing
Data Type             boolean
```

© Inductive Automation

Enable Point Deletion

If true, allow the user to temporarily remove samples from chart. This is used to remove samples that are known to out of control before calculating control limits. The sample that have been removed are not removed from the database and can be restored by selecting the Restore Points menu item.

```
Scripting name        enablePointDeletion
Data Type             boolean
```

Enable Control Limit Editing

If true, allow the user to calculate and set new control limit values.

```
Scripting name        enableControlLimitEditing
Data Type             boolean
```

Events

This component has standard Ignition events.

(none)

Methods

showSetLimitPanel(chartName)

Causes the calculate and set control limit dialog to be shown.

parameters

chartName — Which chart to show the control limit dialog for. Available c "Primary".

```
Data Type             String
```

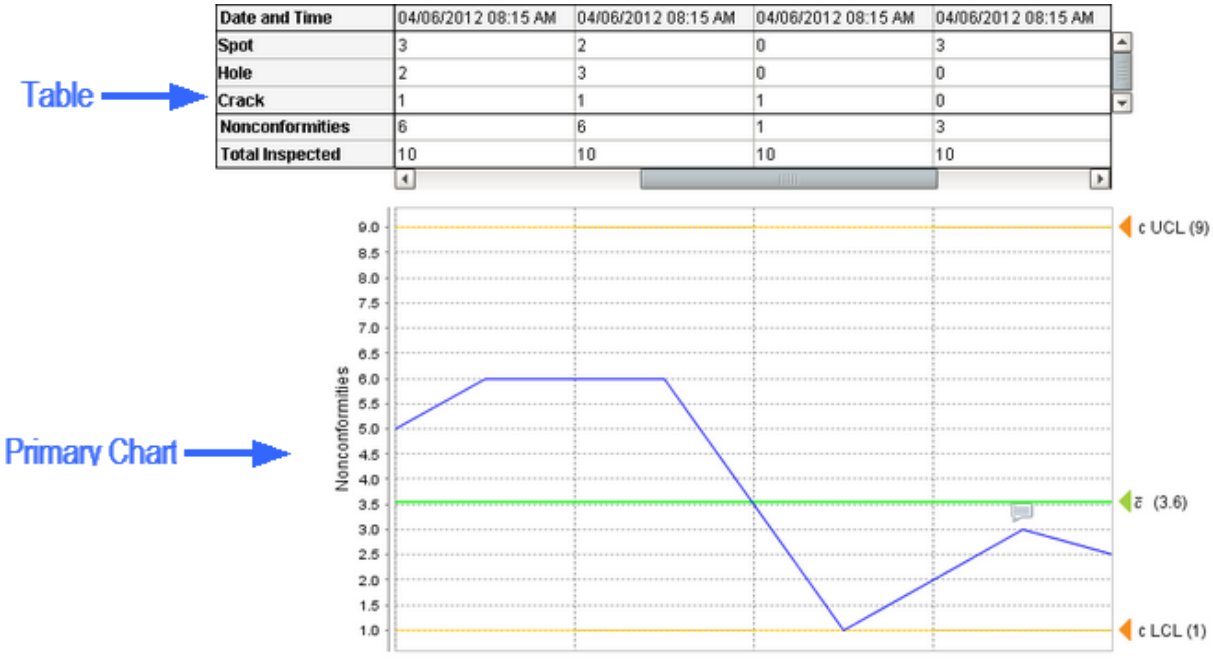
returns nothing

3.4.2.13 C-Chart

Description

The Number of Nonconformities (c) control chart is used to display SPC results that have nonconformities counts for each sample. It does not retrieve SPC results from the SPC module so it must be used with either the SPC Selector or the SPC Controller components that do. Only SPC results with c chart SPC Data Format specified will be displayed.

© Inductive Automation



C Control Chart

Through the use of the properties listed below, the appearance and functionality of this component can modified as desired.

Properties

This component has standard Ignition properties with the addition of the following properties:

Data Properties

SPC Results

Bind this property to the SPC Results property of the SPC Controller or SPC Selector components. Ther SPC Results contain all the information so that the control charts can display the results which is not possible with the data set alone.

```
Scripting name        spcResults
Data Type             SPCResults
```

SPC Data

This property holds the SPC data and includes columns for the calculated values based on the SPC Data Format and selected attribute, control limits and signals.

```
Scripting name        spcData
Data Type             Dataset
```

Measurement Count

This property represents the number of measurements for each sample in the SPC results.

```
Scripting name        measurementCount
Data Type             int
```

© Inductive Automation

User This property can optionally be set to override the current user logged in. It is used when notes or assignable causes are added or modified.

```
Scripting name        user
Data Type             String
```

Background Color The background color.

```
Scripting name        backgroundColor
Data Type             Color
```

Definition Name The sample definition to used when building SPC results.

```
Scripting name        definitionName
Data Type             String
```

No Data Message Text to display if not data is available to show in the control chart.

```
Scripting name        noDataMessage
Data Type             String
```

No Data Foreground The foreground color to display the no data message.

```
Scripting name        noDataForeground
Data Type             Color
```

No Data Font The font to display the no data message.

```
Scripting name        noDataFont
Data Type             Font
```

Table Properties

Show Table If true, the table containing measurement and calculated values will be shown at the top of the control chart.

```
Scripting name        showTable
Data Type             boolean
```

Min Visible Samples The minimum number of sample to show on the control chart. If more than the minimum visible samples exist in the SPC results, then a horizontal scroll bar will appear and allow the user to scroll back to earlier samples.

```
Scripting name        minVisibleSamples
Data Type             int
```

© Inductive Automation

Min Visible Measurements The minimum number of measurements to show in the table of the control chart. If more than the minimum visible measurements exist in the SPC results, then a vertical scroll bar will appear and allow the user to through all measurements.

```
Scripting name        minVisibleMeasurements
Data Type             int
```

Column Width The width of the table column for each sample. The charts will be expanded to match the column width.

```
Scripting name        columnWidth
Data Type             int
```

Row Height The height of the table rows.

```
Scripting name        rowHeight
Data Type             int
```

Date Background The background color of the sample date row.

```
Scripting name        dateBackground
Data Type             Color
```

Date Foreground The foreground color of the sample date values.

```
Scripting name        dateForeground
Data Type             Color
```

Date Font The font to display the sample date values.

```
Scripting name        dateFont
Data Type             Font
```

Date Format The date formatting pattern to display the sample dates.

```
Scripting name        dateFormat
Data Type             String
```

Label Background The background color of the labels.

```
Scripting name        labelBackground
Data Type             Color
```

Label Foreground The foreground color of the labels.

```
Scripting name        labelForeground
Data Type             Color
```

© Inductive Automation

Label Font The font to display the labels.

```
Scripting name        labelFont
Data Type             Font
```

Data Background The background color of the measurement data values.

```
Scripting name        dataBackground
Data Type             Color
```

Data Foreground The foreground color of the measurement data values.

```
Scripting name        dataForeground
Data Type             Color
```

Data Font The font to display the measurement values.

```
Scripting name        dataFont
Data Type             Font
```

Calc Background The background color of the calculated data values.

```
Scripting name        calcBackground
Data Type             Color
```

Calc Foreground The foreground color of the calculated data values.

```
Scripting name        calcForeground
Data Type             Color
```

Calc Font The font to display the calculated values.

```
Scripting name        calcFont
Data Type             Font
```

Chart Properties

Show Primary Chart If true, the primary chart will appear.

```
Scripting name        showPrimaryChart
Data Type             boolean
```

Right Axis Width The width of the right chart axis in pixels.

```
Scripting name        rightAxisWidth
Data Type             int
```

© Inductive Automation

Limit Dialog Horizontal Offset

The horizontal, or x, position to display the set control limit dialog box.

```
Scripting name        limitDialogHorizontalOffset
Data Type             int
```

Limit Dialog Vertical Offset

The vertical, or y, position to display the set control limit dialog box.

```
Scripting name        limitDialogVerticalOffset
Data Type             int
```

Vertical Grid Line Color

The color of the chart vertical grid lines.

```
Scripting name        verticalGridLineColor
Data Type             Color
```

Show Vertical Grid Lines

If true, show the vertical grid lines in the charts.

```
Scripting name        showVerticalGridLines
Data Type             boolean
```

Horizontal Grid Line Color

The color of the chart horizontal grid lines.

```
Scripting name        horizontalGridLineColor
Data Type             Color
```

Show Horizontal Grid Lines

If true, show the horizontal grid lines in the charts.

```
Scripting name        showHorizontalGridLines
Data Type             boolean
```

Primary Chart Background

The background color of the primary chart.

```
Scripting name        primaryChartBackground
Data Type             Color
```

Show Notes

If true, show the note icon next to the chart point for any samples that have notes or assignable causes.

```
Scripting name        showNotes
Data Type             boolean
```

Note Image

The image to display next to the chart point for any samples that have notes or assignable causes.

```
Scripting name        noteImage
Data Type             Image
```

Enable Note Editing

If true, allow the user to add and edit notes and assignable causes.

```
Scripting name        enableNoteEditing
Data Type             boolean
```

© Inductive Automation

Enable Point Deletion

If true, allow the user to temporarily remove samples from chart. This is used to remove samples that are known to out of control before calculating control limits. The sample that have been removed are not removed from the database and can be restored by selecting the Restore Points menu item.

```
Scripting name        enablePointDeletion
Data Type             boolean
```

Enable Control Limit Editing

If true, allow the user to calculate and set new control limit values.

```
Scripting name        enableControlLimitEditing
Data Type             boolean
```

Events

This component has standard Ignition events.

(none)

Methods

showSetLimitPanel(chartName)

Causes the calculate and set control limit dialog to be shown.

parameters

chartName — Which chart to show the control limit dialog for. Available c "Primary".

```
Data Type             String
```

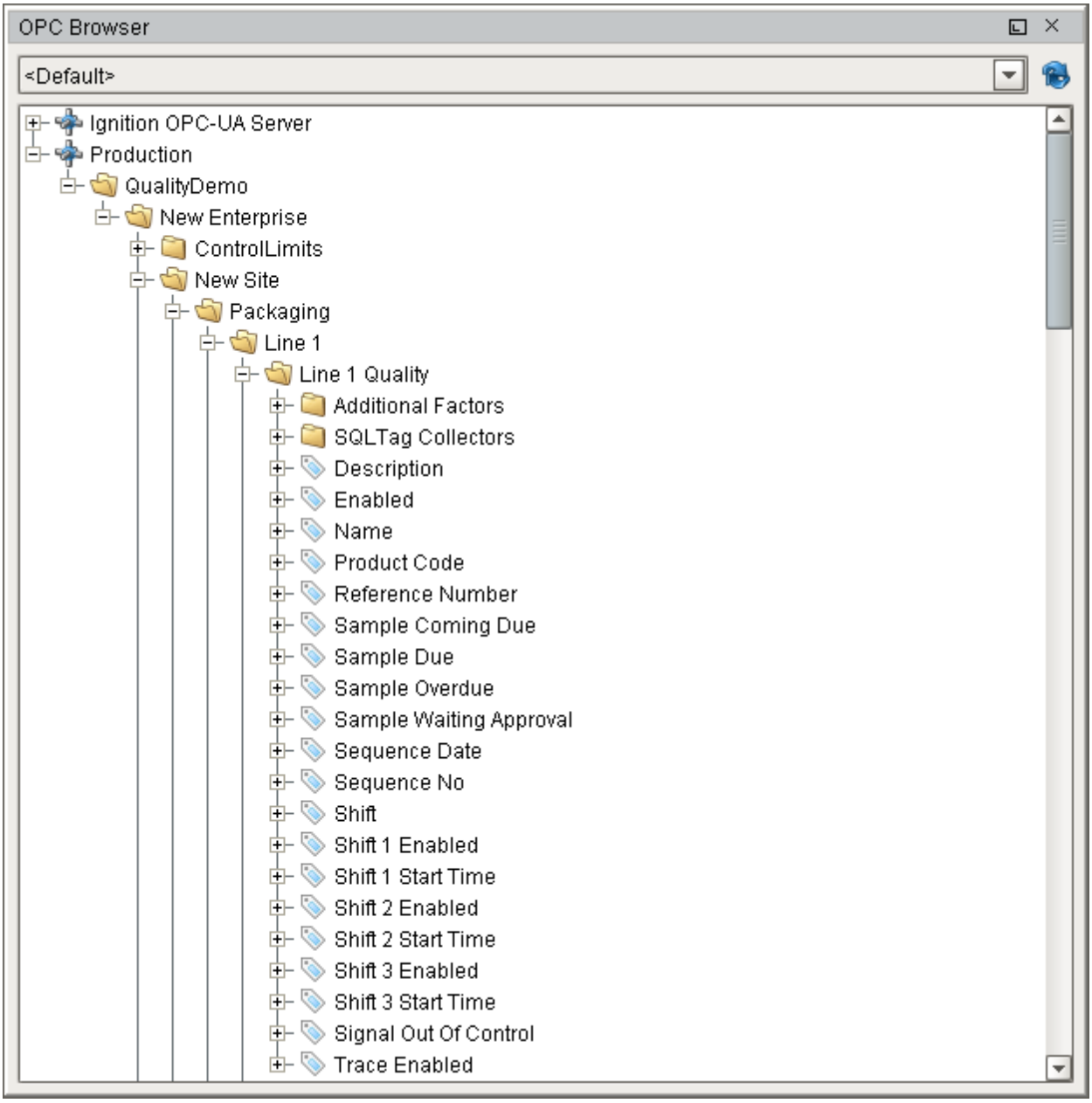
returns nothing

3.5 Quality OPC Values

The production model is defined in the Ignition designer and contains your production *areas*, *lines* and *locations*. Runtime access into configuration and current state of the production model is available through the Production OPC Server. It is added automatically when the SPC Module is installed. When the production items are added, removed or modified, the changes will be reflected in the Production OPC Server when the project is saved in the designer.

Below is a part of the values available to read, and in some cases write to, for the demo project.

© Inductive Automation

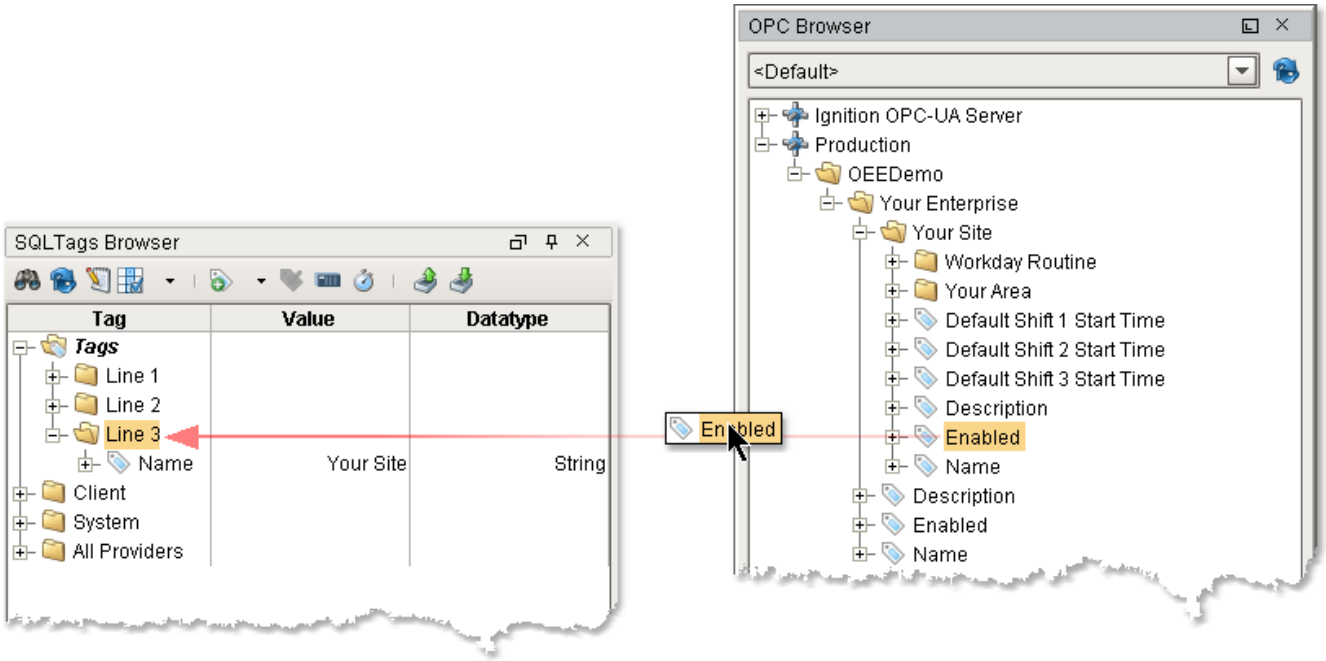


Demo OPC Values

3.5.1 Using OPC Values_2

The SPC configuration settings and runtime values are available for use in Ignition windows, transaction groups, scripting, etc. Before values from the Production OPC Server can be used, they must be added to the Ignition SQLTags. This is done in the designer by selecting the SQLTags Browser and clicking on the icon. This will cause the OPC Browser to appear. Next, drill down in the **Production** node within the OPC Browser. Drag the desired Production OPC Values over to the SQLTags Browser as depicted below.

© Inductive Automation



Add Production OPC Server Values to SQLTags

Important:
When writing to OPC values that are related to production model settings, the new value is not retained upon restarting. This is because production model settings are saved in the Ignition project and is only saved when done so in the designer.

3.5.2 SPC OPC Value Reference

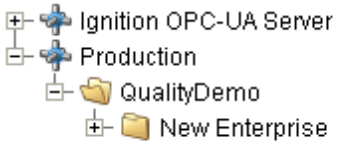
This references details the OPC values and child folders for node types that appear when browsing the Production OPC Server. For each property, the Ignition data type is listed and if it is read only. The Ignition data types correspond to the data types that are available for SQLTags.

Within this reference, the "Read Only" means that the OPC value cannot be written to through the OPC Production Server. It can only be set in the designer or it is a calculated value. Trying to write to a read only property will result in an error message being shown.

3.5.2.1 Project

Description

Each project within Ignition has its own production model. The first node(s) under the main ***Production*** node represent the Ignition project(s). Their names are the same as the project name. The image below represents the OEEDemo project.



Project

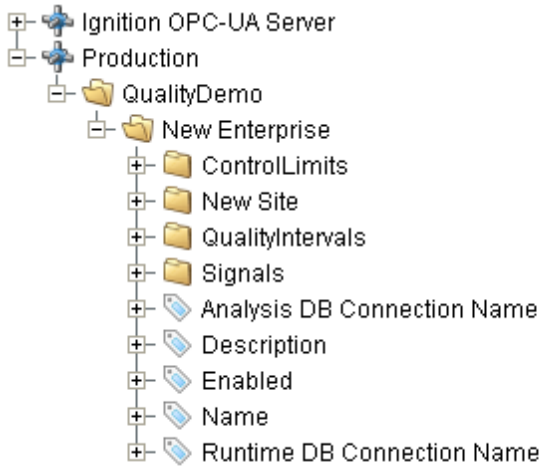
Child Folders

Enterprise One folder will exist for each *Enterprise* that has been configured in the Ignition Designer. The folder can be opened to view all values within the *enterprise*.

3.5.2.2 Enterprise

Description

The *enterprise* folder contains some properties associated with the *enterprise* and a folder for each production *Site* within it. The name is the same as the *enterprise* name that is configured in the designer. The image below represents the "New Enterprise" of the QualityDemo project.



Enterprise

Child Folders

Site One folder will exist for each *Site* that has been configured in the Ignition Designer. The folder can be opened to view all values within the *site*.

Control Limits This is the parent folder that holds all of the control limits.

Signals This is the parent folder that holds all of the signals.

Intervals This is the parent folder that holds all of the intervals.

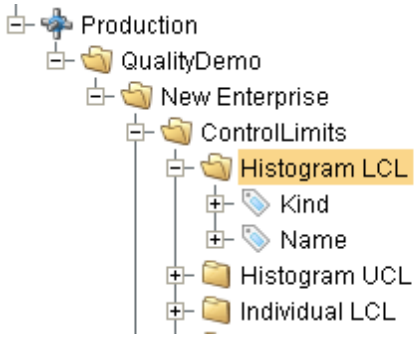
Properties

Description	Optionally, this property can be set to a description for the *enterprise*. It is not used by the SPC Module other than for reference.	String
Enabled	This reflects the *enterprise* Enabled property in the Designer. If the *enterprise* Enabled is set to true, then the SPC module will perform calculations for the *enterprise* and all *sites*, *areas*, *lines* and locations within it. If this property is set to false, then none of the *sites*, *areas*, *lines* or *locations* will have calculations performed.	Boolean
Name	This reflects the name of the *enterprise* that is set in the designer.	String Read Only
Analysis DB Connection Name	This reflects the Analysis Database setting in the MES section in the Ignition Gateway.	String Read Only
Runtime DB Connection Name	This reflects the Runtime Database setting in the MES section in the Ignition Gateway.	String Read Only

3.5.2.2.1 Control Limits

Description

The control limits folder contains a folder for each control limit. The name of each folder is the same as the control limit name that is configured in the designer. The image below represents the "Histogram LCL" control limit of the QualityDemo project.



Control Limits

Properties

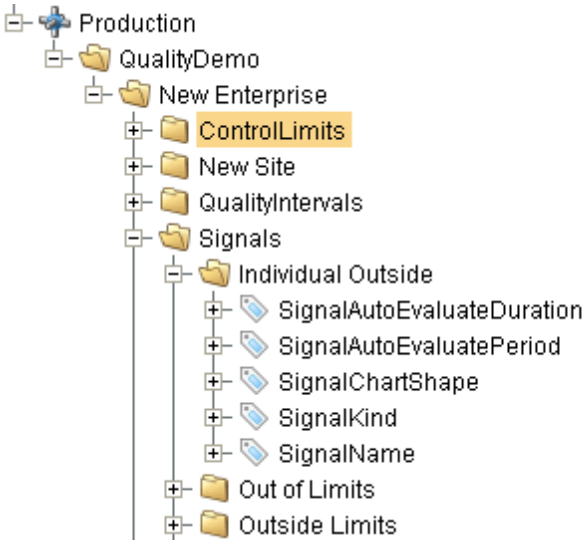
Kind	The ordinal value of the kind of control chart that the control limit is associated with. See ControlLimitKindTypes for more information.	int Read Only
Name	This reflects the name of the control limit that is configured in the designer.	String Read Only

3.5.2.2.2 Signals

Description

The signals folder contains a folder for each signal. The name of each folder is the same as the signal name that is configured in the designer. The image below represents the "Individual Outside" signal of the QualityDemo project.

© Inductive Automation



Out of Control Signals

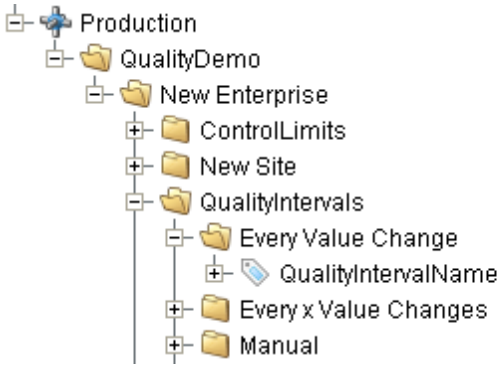
Properties

Kind	The ordinal value of the kind of control chart that the signal is associated with. See `SignalKindTypes for more information.`	int Read Only
SignalName	This reflects the name of the signal that is configured in the designer.	String Read Only
SignalAutoEvaluatePeriod	This reflects the ordinal value of the evaluation time period of the SignalAutoEvaluateDuration value. See `SignalAutoEvaluatePeriodTypes for more information.`	int Read Only
SignalAutoEvaluateDuration	This reflects the duration to use when automatically evaluating sample data for a location for this signal.	int Read Only
SignalChartShape	This reflects the ordinal value of the shape to display in the control charts when a sample is out of control for this signal. See `SPCChartShapeTypes for more information.`	int Read Only

3.5.2.2.3 Intervals

Description

The quality intervals folder contains a folder for each interval. The name of each folder is the same as the interval name that is configured in the designer. The image below represents the "Every Value Change" interval of the QualityDemo project.



Sampling Intervals

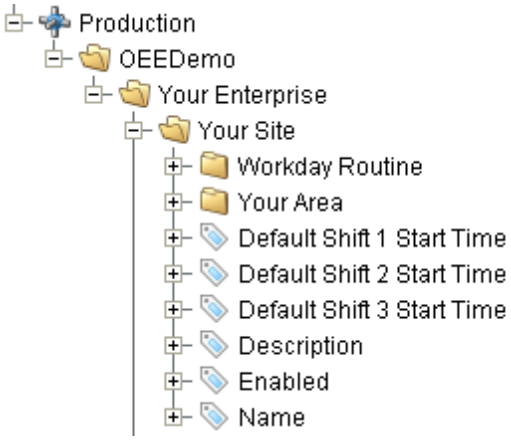
Properties

QualityIntervalName	This reflects the name of the interval that is configured in the designer.	String Read Only

3.5.2.3 Site

Description

The *site* folder contains some properties associated with the production *site* and a folder for each production *area* within it. The name is the same as the *site* name that is configured in the designer. The image below represents the "Your Site" of the QualityDemo project.



Site

Child Folders

Area	One folder will exist for each area that has been configured in the Ignition Designer. The folder can be opened to view all values within the *area*.

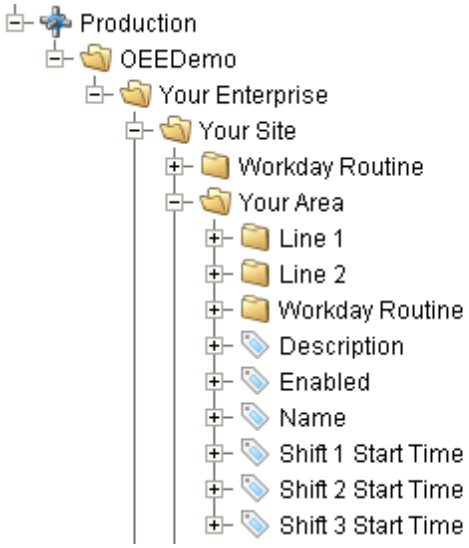
Properties

Description	Optionally, this property can be set to a description for the *site*. It is not used by the OEE Downtime and Scheduling Module other than for reference.	String
Enabled	This reflects the *site* Enabled property in the Designer. If the *site* Enabled is set to true, then the OEE Downtime and Scheduling module will perform calculations for the *site* and all *areas*, *lines* and *cells* within it. If this property is set to false, then none of the *areas*, *lines* or *cells* will have calculations performed.	Boolean
Name	This reflects the name of the *site* that is set in the designer.	String Read Only
Default Shift 1 Start Time	This reflects the *site* Default Shift 1 Start Time property in the Designer. See Site Configuration for more details.	DateTime Read Only
Default Shift 2 Start Time	This reflects the *site* Default Shift 2 Start Time property in the Designer. See Site Configuration for more details.	DateTime Read Only
Default Shift 3 Start Time	This reflects the *site* Default Shift 3 Start Time property in the Designer. See Site Configuration for more details.	DateTime Read Only

3.5.2.4 Area

Description

The *area* folder contains some properties associated with the production *area* and a folder for each production *line* within it. The name is the same as the *area* name that is configured in the designer. The image below represents the "Your Area" of the QualityDemo project.



Area

Child Folders

Line One folder will exist for each *Line* that has been configured in the Ignition Designer. The folder can be opened to view all values within the *line*.

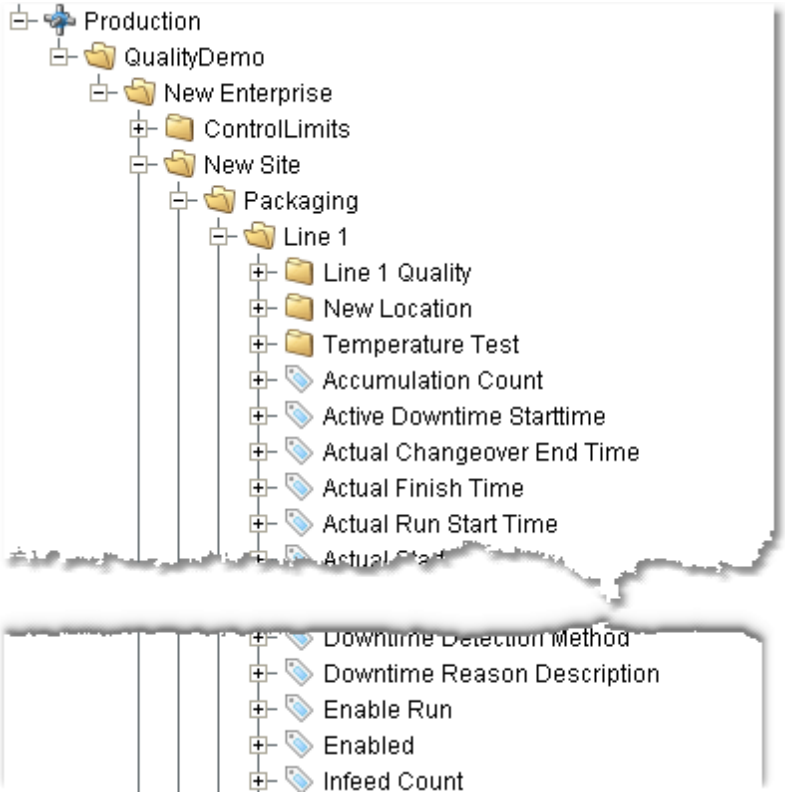
Properties

Description	Optionally, this property can be set to a description for the *area*. It is not used by the OEE Downtime and Scheduling Module other than for reference.	String
Enabled	This reflects the *site* Enabled property in the Designer. If the *area* Enabled is set to true, then the OEE Downtime and Scheduling module will perform calculations for the *area* and all *lines* and cell within it. If this property is set to false, then none of the *lines* or cells will have calculations performed.	Boolean
Name	This reflects the name of the *area* that is set in the designer.	String Read Only
Shift 1 Start Time	The current Shift 1 Start Time time for the production *area*. If the associated Shift 1 Start Time property for the *area* in the designer is set to *Inherit From Parent*, this will be the time defined for the parent production *site*. See Area Configuration for more details.	DateTime Read Only
Shift 2 Start Time	The current Shift 2 Start Time time for the production *area*. If the associated Shift 2 Start Time property for the *area* in the designer is set to *Inherit From Parent*, this will be the time defined for the parent production *site*. See Area Configuration for more details.	DateTime Read Only
Shift 3 Start Time	The current Shift 3 Start Time time for the production *area*. If the associated Shift 3 Start Time property for the *area* in the designer is set to *Inherit From Parent*, this will be the time defined for the parent production *site*. See Area Configuration for more details.	DateTime Read Only

3.5.2.5 Line

Description

The *line* folder contains some properties associated with the production *line* and a folder for each production location within it. The name is the same as the *line* name that is configured in the designer. The image below represents the "Line 1" of the QualityDemo project.



Line

Child Folders

Location One folder will exist for each ***Location*** that has been configured under the line in the Ignition Designer. The folder can be opened to view all values within the location.

Properties

Only the properties that may be useful for the SPC module are shown below. See OEE DT Line for more properties.

Enabled	This reflects the *line* Enabled property in the Designer. If the *line* Enabled is set to true, then the OEE Downtime and Scheduling module will perform calculations for the all cells within it. If this property is set to false, then none of the cells will have calculations performed.	Boolean
Name	This reflects the name of the *line* that is set in the designer.	String Read Only
Product Code	The current product code being run on the *line*. Typically, this is controlled by the functionality of the operator screen, but it can also be handled programmatically. It should only be changed when *Enable Run* is false.	String
Product Code Description	The description for the current *Product Code.*	String Read Only
Running	This value will be true if a production run is started and production *line* is running.	Boolean Read Only
Sequence No	A number that is 0 at the beginning of a production run and increments at the beginning of every shift.	Int4 Read Only
Shift	The current shift based on the shift start times configured for the production *line*.	Int4 Read Only
Shift 1 Enabled	The current Shift 1 enabled state for the production *line*. It reflects the Shift 1 Enabled property for the *line* in the designer. The initial value of this property is determined by the Shift 1 Initial Enabled State property for the production *line* in the designer. See Line Configuration for more details. It can be changed from the initial value.	Boolean
Shift 1 Start Time	The current Shift 1 Start Time time for the production *line*. If the associated Shift 1 Start Time property for the *line* in the designer is set to *Inherit From Parent*, this be the time defined for the parent production *area*. See Line Configuration for more details.	DateTime Read Only
Shift 2 Enabled	The current Shift 2 enabled state for the production *line*. It reflects the Shift 2 Enabled property for the *line* in the designer. The initial value of this property is determined by the Shift 2 Initial Enabled State property for the production *line* in the designer. See Line Configuration for more details. It can be changed from the initial value.	Boolean

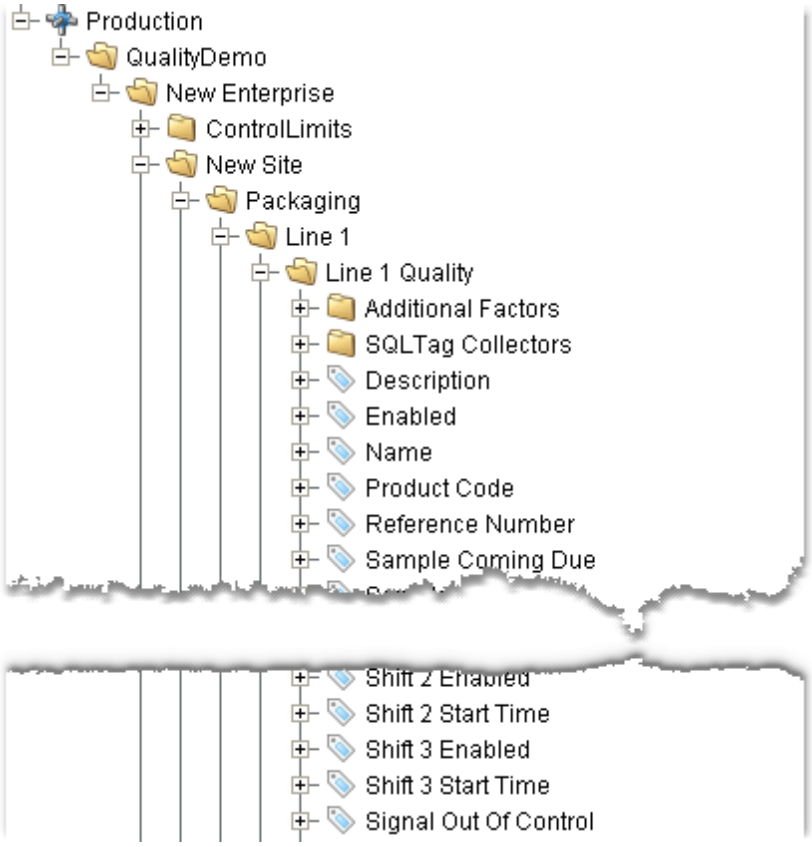
© Inductive Automation

Shift 2 Start Time	The current Shift 2 Start Time time for the production *line*. If the associated Shift 2 Start Time property for the *line* in the designer is set to *Inherit From Parent*, this be the time defined for the parent production *area*. See Line Configuration for more details.	DateTime Read Only
Shift 3 Enabled	The current Shift 3 enabled state for the production *line*. It reflects the Shift 3 Enabled property for the *line* in the designer. The initial value of this property is determined by the Shift 3 Initial Enabled State property for the production *line* in the designer. See Line Configuration for more details. It can be changed from the initial value.	Boolean
Shift 3 Start Time	The current Shift 3 Start Time time for the production *line*. If the associated Shift 3 Start Time property for the *line* in the designer is set to *Inherit From Parent*, this be the time defined for the parent production *area*. See Line Configuration for more details.	DateTime Read Only
Work Order	The current work order number for the current production run.	String Read Only

3.5.2.6 Location

Description

The location folder contains properties associated with the production location. The production location can reside under a production line or directly under a production area. The name is the same as the location name that is configured in the designer. The image below represents the **Line 1 Quality** location of the QualityDemo project.



Location

© *Inductive Automation*

Child Folders

Additional Factors	Contains all of the additional factor entries that have been configured for the production *location*.
SQLTag Collectors	Contains all of the tag collector entries that have been configured for the production *location*.

Properties

Description	Optionally, this property can be set to a description for the location. It is not used by the SPC Module other than for reference.	String
Enabled	If Enabled is set to true, then the SPC module will perform calculations and enable tag collectors for the location.	Boolean
Name	This reflects the name of the location that is set in the designer.	String Read Only
Product Code	This reflects the product code currently assigned to this location.	String Read Only
Reference Number	This reflects the reference number currently assigned to this location. The reference number is optional and can represent anything that samples will be tracked by except for the product code.	String Read Only
Sample Coming Due	If true, a sample is coming due for this location.	Boolean Read Only
Sample Due	If true, a sample is due for this location.	Boolean Read Only
Sample Overdue	If true, a sample is overdue for this location.	Boolean Read Only
Sample Waiting Approval	If true, an unapproved sample is waiting to be approved for this location.	Boolean Read Only
Sequence Date	The date and time that the current shift started. This is used for retrieving results based on a production day and not days that are split at midnight.	Date Read Only
Sequence No	A number that is 0 at the beginning of a production run and increments at the beginning of every shift.	Int4 Read Only
Shift	The current shift based on the shift start times configured for the production *location*.	Int4 Read Only
Shift 1 Enabled	The current Shift 1 enabled state for the production location. It reflects the Shift 1 Enabled property for the *location* in the designer. The initial value of this property is determined by the Shift 1 Initial Enabled State property for the production *location* in the designer. It can be changed from the initial value.	Boolean
Shift 1 Start Time	The current Shift 1 Start Time time for the production *location*. If the associated Shift 1 Start Time property for the *location* in the designer is set	DateTime

© Inductive Automation

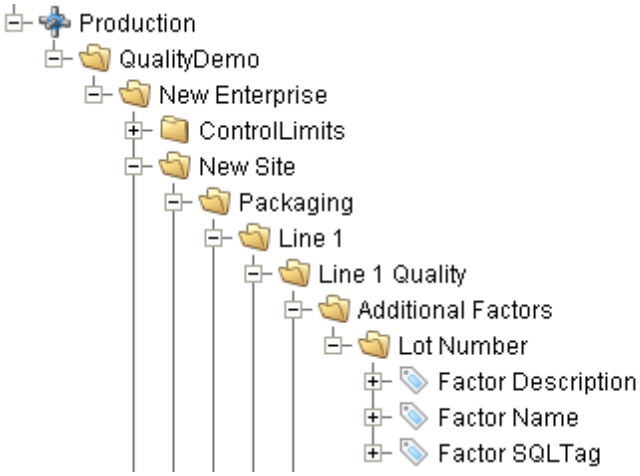
	to *Inherit From Parent*, this can be the time defined for the parent production *area or line*.	
Shift 2 Enabled	The current Shift 2 enabled state for the production location. It reflects the Shift 2 Enabled property for the *location* in the designer. The initial value of this property is determined by the Shift 2 Initial Enabled State property for the production *location* in the designer. It can be changed from the initial value	Boolean
Shift 2 Start Time	The current Shift 2 Start Time time for the production *location*. If the associated Shift 2 Start Time property for the *location* in the designer is set to *Inherit From Parent*, this can be the time defined for the parent production *area or line*.	DateTime
Shift 3 Enabled	The current Shift 3 enabled state for the production location. It reflects the Shift 3 Enabled property for the *location* in the designer. The initial value of this property is determined by the Shift 3 Initial Enabled State property for the production *location* in the designer. It can be changed from the initial value.	Boolean
Shift 3 Start Time	The current Shift 3 Start Time time for the production *location*. If the associated Shift 3 Start Time property for the *location* in the designer is set to *Inherit From Parent*, this can be the time defined for the parent production *area or line*.	DateTime
Signal Out Of Control	If true, at least one signal associated with this location is out of control.	Boolean Read Only
Trace Enabled	If true, a product code has been assigned to this location and is considered as actively processing.	Boolean Read Only

3.5.2.6.1 Additional Factors

Description

The additional factors folder contains a folder for each additional factor within it. The name of each folder is the same as the additional factor name that is configured in the designer. The image below represents the "Line 1 Quality" additional factors of the QualityDemo project.

© *Inductive Automation*



Additional Factors

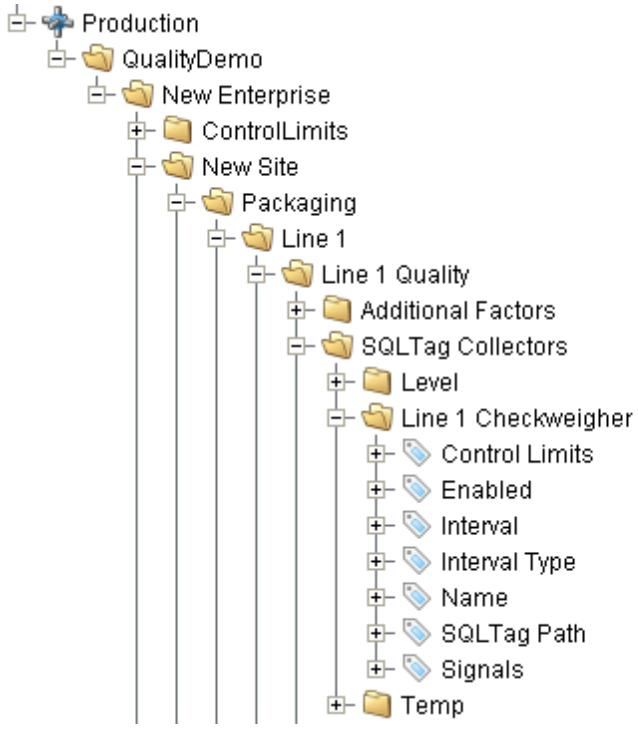
Properties

Factor Description	Optionally, this property can be set to a description for the additional factor. It is not used by the SPC Module other than for reference.	String
Factor Name	This reflects the name of the additional factor that is configured in the designer.	String Read Only
Factor SQLTag	This reflects the Factor SQLTag setting that the additional factor is configured for in the designer. It is the name of the SQLTag to read the factor value from.	String Read Only

3.5.2.6.2 Tag Collectors

Description

The SQLTag Collectors folder contains a folder for each tag collector within it. The name of each folder is the same as the tag collector name that is configured in the designer. The image below represents the "Line 1 Checkweigher" tag collector of the QualityDemo project.



Tag Collectors

Properties

Name	This reflects the name of the tag collector that is configured in the designer.	String Read Only
SQLTag Path	This reflects the SQLTag path that is configured in the designer from which the sample measurement data is read.	String Read Only
Enabled	If true, the tag collector will automatically read the value from the associated tag and create samples based on the interval.	Boolean
Interval Type	This reflects the sample interval to use with this tag collector as configured in the designer.	String Read Only
Interval	This reflects the sample interval value as configured in the designer. The meaning depends on the interval type. See Intervals for more information.	Double Read Only
Control Limits	This reflects the control limits that will be calculated during signal evaluation as configured in the designer.	String Read Only
Signals	This reflects the out of control signal(s) to be evaluated automatically when a new sample is collected.	String Read Only

© Inductive Automation

3.6 Scripting

3.6.1 Production Location Events

The following events are by location, which allows for the changing of default handling samples and detection of out of control signals by individual location. Individual handling based on the of sample or other criteria, must be done in the script.

In situations where the default handling does not fit the production environment requirements, these events are flexible enough to allow a method to implement exactly what is needed.

3.6.1.1 Before Sample Updated Event

Before a new sample is added or an existing sample is updated to the database, any script in this event is run. This includes samples that have been scheduled with no measurement data. It is provided to allow for the addition of more information, performing other actions or preventing the saving of the sample.

event properties:

- getSample() - Sample
  Returns the new or updated sample. (See Sample section more information).

- setCancelUpdate(boolean cancelUpdate)
  Used to prevent the sample from being added or updated. The default is false, meaning the sample will be added or updated. It is provided to override the default adding or updating of samples and should be used with caution.

- isCancelUpdate() - boolean
  Returns the current state of the cancel update flag.

Example:

```
#Add 1 to an unrelated SQLTag value
val = system.tag.getTagValue('[Default]Quality\Test\Before Sample Updated')
val = val + 1
system.tag.writeToTag('[Default]Quality\Test\Before Sample Updated', val, 1)

#Get the sample from the event
sample = event.getSample()

#Access the additional factors from the sample
addlFactors = sample.getAllAddlFactors()
if len(addlFactors) > 0:
   print "%d additional factors exist." % (len(addlFactors))

   print "val = %d, sampleUUID = %s" % (val, sample.getSampleUUID())
```

3.6.1.2 After Sample Updated Event

After a new sample is added or an existing sample is updated to the database, any script in this event is run. This includes samples that have been scheduled with no measurement data. It is provided to allow for the performance of other actions when sample information changes.

event properties:

- getSample() - Sample

Returns the new or updated sample. (See Sample section more information).

Example:

```
#Add 1 to an unrelated SQLTag value
val = system.tag.getTagValue('[Default]Quality\Test\After Sample Updated')
val = val + 1
system.tag.writeToTag('[Default]Quality\Test\After Sample Updated', val, 1)
```

3.6.1.3 Sample Approval Updated Event

After the sample approval state has been updated, any script in this event is run. This includes samples that are set for automatic approval. It is provided to allow for the performance of other actions when sample approval state changes.

event properties:

- getSample() - Sample
  Returns the sample for which the approval state changed. (See Sample section more information).

- isApproval() - boolean
  Returns true if the sample has been approved.

- isUnapproval() - boolean
  Returns true if the sample has been unapproved.

Example:

```
#Add 1 to an unrelated SQLTag value
val = system.tag.getTagValue('[Default]Quality\Test\Sample Approval Updated')
val = val + 1
system.tag.writeToTag('[Default]Quality\Test\Sample Approval Updated', val, 1)
```

3.6.1.4 Sample Coming Due Event

When a sample due state changes to COMING_DUE, any script in this event is run. It is provided to allow for the performance of other actions, such as alerts, when sample is coming due.

event properties:

- getSample() - Sample
  Returns the sample that just became due. (See Sample section more information).

- getState() - SampleDueStateTypes
  Returns the current sample due state (See Sample Due State Types for more information).

Example:

3.6.1.5 Sample Due Event

When a sample due state changes to DUE, any script in this event is run. It is provided to allow for the performance of other actions, such as alerts, when sample is due.

event properties:

- getSample() - String
  Returns the sample that just became due. (See Sample section more information).

- getState() - SampleDueStateTypes
  Returns the current sample due state (See Sample Due State Types for more information).

© Inductive Automation

Example:

3.6.1.6 Sample Interval Event

Evaluates the sample interval and determines when to create a new sample.

event properties:

- getDefUUID() - String
  Returns the definition UUID for this sample.
- getInterval() - Double
  Returns the defined interval of this sample .
- getDuration() - Double
  Returns the number of minutes needed to take a sample.
- getTag() - String
  Returns the tag associated with this sample.
- getComingDueMin() - Double
- getOverDueMin() - Double
- getSecSinceLastSampleScheduled() - Integer
  Returns the seconds since the last sample was scheduled .
- getSecSinceLastSampleTaken() - Integer
  Returns the seconds since the last sample was taken.
- getProductCode() - String
  Returns the product code associated with this sample.
- getRefNo() - String
  Returns the reference number associated with this sample.
- getTraceEnabled() - Boolean
- getTraceStartedAt() - Date
- getElapsedSeconds() - Integer
  Returns the .
- getTraceEndedAt() - Date
- getSequenceDate() - Date

© Inductive Automation

Returns the sequence date of the sample. Sequence date is the date representing the start of the current shift.

- getSequenceNo() - Integer
  Returns the sequence number of the sample. Sequence number is the sequential number of shifts from the start of the production run.
- getShift() - Integer
  Returns the shift number.
- getValueChangeCount() - Integer
  Returns the number of time the associated value has changed.
- getValueChangedTimeStamp() - Date
  Returns the date time the value changed.
- getValue() - Object
  Returns the value of the sample.

- isTracedStartedEvent() - Boolean
- isTracedEndedEvent() - Boolean
- isShiftChangeEvent() - Boolean
- getCreateSample() - Boolean
- setCreateSample(createSample Boolean)
- getScheduleStart() - Date
- setScheduleStart(Date)
- getScheduleFinish() - Date
- setScheduleFinish(Date)
- getRefresh() - Boolean

© Inductive Automation

- setRefresh(refresh Boolean)

3.6.1.7 Sample Overdue Event

When a sample due state changes to OVERDUE, any script in this event is run. It is provided to allow for the performance other actions, such as alerts, when sample is overdue.

event properties:

- getSample() - Sample
  Returns the sample that just became due. (See Sample section more information).

- getState() - SampleDueStateTypes
  Returns the current sample due state (See Sample Due State Types for more information).

Example:

3.6.1.8 Sample Waiting Approval Event

When a sample due state changes to WAITING_APPROVAL, any script in this event is run. It is provided to allow for the performance of other actions, such as alerts, when sample is awaiting approval.

event properties:

- getSample() - Sample
  Returns the sample that just became due. (See Sample section more information).

- getState() - SampleDueStateTypes
  Returns the current sample due state (See Sample Due State Types for more information).

Example:

3.6.1.9 Signals Evaluated Event

When sample data changes, all of the out of control signals associated with it will be evaluated. After each attribute for each definition has been evaluated, any script in this event is run. It is provided to allow for special handling to override out of control conditions as described below. A preferred alternative is to implement the desired results in a Interval (See Intervals for more information).

event properties:

- getDefUUID() - String
  Returns the definition UUID that was evaluated.. (See Sample Definition section more information).

- setIgnoreOutOfControl(boolean ignoreOutOfControl)
  Used to override and ignore an out of control condition.

- setForceOutOfControl(boolean forceOutOfControl)
  Used to force an out of control condition.

- getEvaluationResults() - List<SignalEvaluationResults>
  Returns a list of evaluation results. When sample data is updated for a location - sample definition combination, all of the selected signals are evaluated. This occurs for each attribute within the sample definition.

© Inductive Automation

Example:

If sample definition *viscosity* has an allowable location *processing,* has two attributes of *cold viscosity* and *temperature, a*nd *signal rule 1* and *signal rule 2* are selected*,* then when a sample is added or updated, *cold viscosity* for *signal rule 1*, *cold viscosity* for *signal rule 2*, *temperature* for *signal rule 1* and *temperature* for *signal rule 2* are all evaluated. The outcome for each combination is a item within the evaluation results returned from the getEvaluationResults() function.

3.6.1.10 Signal Evaluation Results

This object holds the evaluation results for a attribute signal combination.

event properties:

- getSignalName() - String
  Returns the name of the signal associated with this result.

- getAttributeName() - String
  Returns the name of the attribute associated with this result.

- getViolatingSampleDate() - Date
  Returns the date of the most recent sample that is in violation of the signal.

- getLastSampleDate() - Date
  Returns the date of the last approved sample. This can be used in combination to determine if the last approved sample caused the signal violation.

- isSignalViolation() - boolean
  Returns true if the signal - attribute combination are in violation.

- isEvaluationError() - boolean
  Returns true if a error occurred during the signal evaluation.

- hasMessage() - boolean
  Returns true if a message exists.

- getMessage() - String
  Returns textual description of error encountered during the signal evaluation.

Example:

3.6.1.11 Signal Out of Control Event

When sample data changes, all of the out of control signals associated with it will be evaluated. If an out of control signal changes from “In Control” to “Out of Control”, any script in this event is run. It is provided to allow for the performance of other actions, such as alerts, when an out of control condition occurs.

event properties:

- getDefUUID() - String
  Returns the definition UUID associated with this out of control event. (See Sample Definition section more information).

- getEvaluationResults() -SignalEvaluationResults
  Returns a single evaluation result of the signal - attribute combination that transitioned from in control to out of control.

© Inductive Automation

3.6.1.12 Signal in Control Event

When sample data changes, all of the out of control signals associated with it will be evaluated. If an out of control signal changes from “Out of Control” to “In Control”, any script in this event is run. It is provided to allow for the performance of other actions, such as alerts, when an out of control condition no longer exists.

event properties:

- getDefUUID() - String
  Returns the definition UUID associated with this in control event. (See Sample Definition section more information).

- getEvaluationResults() -SignalEvaluationResults
  Returns a single evaluation result of the signal - attribute combination that transitioned from out of control to in control.

3.6.1.13 Sample Due State Types

UNKNOWN
COMING_DUE
DUE
OVERDUE
WAITING_APPROVAL:

3.6.2 Object Reference

3.6.2.1 Sample

The sample object holds all of the information associated with one sample.

properties:

- getDefUUID() - String
  Returns the definition UUID associated with this sample. (See Sample Definition object for more information).

- getSampleUUID() - String
  Returns the UUID assigned to this sample. A UUID is a universally unique identifier that, once assigned to a sample, will never change. It is also unique in that no two samples will have the same UUID.

- getEnterprise() - String
  Returns the enterprise associated with this sample.

- setEnterprise(String enterprise)
  Sets the enterprise associated with this sample.

- getSite() - String
  Returns the physical production facility associated with this sample.

- setSite(String site)
  Sets the physical production site associated with this sample.

- getArea() - String
  Returns the production area associated with this sample.

© Inductive Automation

- setArea(String area)
  Sets the production area associated with this sample.

- getLine() - String
  Returns the production line associated with this sample. This will be blank if the location the sample is taken is in a production area and not on a production line.

- setLine(String line)
  Sets the production line associated with this sample.

- getLocation() - String
  Returns the location associated with this sample.

- setLocation(String location)
  Sets the production location associated with this sample.

- getLocationPath() - String
  Returns the full location path, including enterprise, site, area, line and location, associated with this sample.

- setLocationPath(String locationPath)
  Sets the enterprise, site, area, line and location from the locationPath parameter.

- getProductCode() - String
  Returns the product code associated with this sample. This is optional and may not apply if tracking quality by product code is not being used for the associated sample definition.

- setProductCode(String productCode)
  Sets the product code associated with this sample.

- getRefNo() - String
  Returns the reference number associated with this sample. This is optional and can be used to track information like batch number, lot number, etc. Additional factors can also be used to track information.

- setRefNo(String refNo)
  Sets the reference number associated with this sample.

- getScheduledStart() - Calendar
  Returns the date and time that this sample is scheduled to be taken. For automatic samplings, this value does not apply and will be equal to None.

- setScheduledStart(Calendar scheduleStart)\
  Sets the date and time that this sample is scheduled to be taken.

- calcScheduledFinish()
  Based on the scheduled start date and time and the duration of time to take this sample, calculates the date and time this sample is scheduled to be complete. The getScheduledFinish () value is updated after calling this function. The duration of time required to take a sample is defined in the sample definition. For automatic samplings, this value does not apply.

© Inductive Automation

- getScheduledFinish() - Calendar
  Returns the date and time that taking this sample is scheduled to be complete. For automatic samplings, this value does not apply and will be equal to None.

- setScheduledFinished(Calendar scheduleFinish)
  Sets the date and time that this sample is scheduled to be completed.

- getSampleTakenDateTime() - Calendar
  Returns the date and time that this sample was taken.

- setSampleTakenDateTime(Calendar sampleTakenDateTime)
  Sets the date and time that this sample was taken.

- getEntryDateTime() - Calendar
  Returns the date and time that this sample was entered.

- setEntryDateTime(Calendar entryDateTime)
  Sets the date and time that this sample was entered.

- getShift() - int
  Returns the shift the sample was taken.

- setShift(int shift)
  Sets the shift the sample was taken.

- getSequenceDate() - Calendar
  Returns the date and time that the shift the sample was taken during started.

- setSequenceDate(Calendar sequenceDate)
  Sets the date and time that the shift the sample was taken during started.

- getApproved() - boolean
  Returns true if this sample has been approved. Depending on the settings in the sample definition, samples may be automatically or manually approved.

- setApproved(boolean approved)
  Set to true to approve this sample.

- getSampleTakenBy() - String
  Returns the person's name who was responsible for taking the sample. By default, this is the person who is logged in when the sample is entered. For automatically recorded samples, this will be "Auto".

- setSampleTakenBy(String sampleTakenBy)
  Sets the person's name who was responsible for taking the sample.

- getApprovedBy() - String
  Returns the person's name who approved this sample. For automatically recorded samples, this will be "Auto".

© Inductive Automation

- setApprovedBy(String approvedBy)
  Sets the person's name who approved this sample.

- getApprovedDateTime() - Calendar
  Returns the date and time that this sample was approved. For automatically approved samples, this will be the same as the getEntryDateTime() value.

- setApprovedDateTime(Calendar approvedDateTime)
  Sets the date and time that this sample was approved.

- getTag() - String
  Returns the optional tag value. This is typically used to assign ownership of which department has the responsibility to take this sample.

- setTag(String tag)
  Sets the tag value. This is typically used to assign ownership of which department has the responsibility to take this sample.

- getNote() - String
  Returns the note associated with this sample. This is the note that may have been entered when the sample was entered. Even though this note can be viewed on the control charts or in analysis, it is not the same as the attribute note entered on the control charts.

- setNote(String note)
  Sets the note associated with this sample.

- getSampleDefinition() - SampleDefinition
  Returns the sample definition associated with this sample. (See Sample Definition object for more information.)

- setSampleDefinition(SampleDefinition definition)
  Sets the sample definition associated with this sample. (See Sample Definition object for more information.)

- isNew() - boolean
  Returns true if this is a newly created sample.

- isModified() - boolean
  Returns true if any properties of this sample have been modified.

attribute properties:

- getAttributeDataType(String attrName) - AttributeDataType
  Returns the attribute data type object for the specified attribute name. The attribute data type information is contained in the sample definition and cannot be changed directly in the sample object.

- getAttributeDefaultValue(String attrName) - Object
  Returns the default value for the specified attribute name. The attribute default value is contained in the sample definition and cannot be changed directly in the sample object.

- getAttributeMinValue(String attrName) - Object

© Inductive Automation

Returns the minimum value for the specified attribute name. The attribute minimum value is contained in the sample definition and cannot be changed directly in the sample object.

- getAttributeMaxValue(String attrName) - Object
  Returns the maximum value for the specified attribute name. The attribute maximum value is contained in the sample definition and cannot be changed directly in the sample object.

measurement properties:

- getAllMeasurements() - List<SampleData>
  Returns the measurements associated with this sample. If a sample has been scheduled but the measurement data has not been recorded, the measurement entries will still exist. In this case, use the sampleDataExists() property to determine if the measurement data has been entered.

- isDataModified() - boolean
  Returns true if any measurement values have been modified.

- isSampleDataValid() - boolean
  Returns true if all of the measurement values are valid.

- sampleDataExists() - boolean
  Returns true if sample data has been entered.

- getSampleData(int measNo, String attrName) - SampleData
  Returns SampleData item for the specified measurement number and attribute.

- getSampleDataValue(int measNo, String attrName) - String
  Returns a measurement value as a string for the specified measurement number and attribute name.

- setSampleData(int measNo, String attrName, String value) - boolean
  Sets a measurement value as a string for the specified measurement number and attribute name. Returns true if successful, otherwise returns false.

- isSampleDataValid() - boolean
  Returns true if the measurements have been entered and are valid.

additional factor properties:

- getAllAddlFactors() - List<SampleAdditionalFactor>
  Returns the list of additional factor values associated with this sample.

- getAddlFactor(String factorName) - SampleAdditionalFactor
  Returns the additional factor object specified by the factorName parameter and associated with this sample. Use this function to get the SampleAdditionalFactor object that can be used to change the value of the additional factor.

3.6.2.2 Sample Data

The sample object holds a sample data object for each attribute and measurement. When a sample object is created, it automatically creates a sample data object based on the sample definition.
For example: If sample definition viscosity has two attributes of cold viscosity and temperature and is

configured for 5 measurements, then the sample will contain 10 sample data objects. Five measurements for cold viscosity and five measurements for temperature.

properties:

- getSampleUUID() - String
  Returns the sample UUID that this sample data object belongs to.

- getAttributeName() - String
  Returns the attribute name this sample data object is associated with.

- getMeasNo() - int
  Returns the measurement number this sample data object is associated with.

- getAttrDataType() - AttributeDataType
  Returns attribute data type of this sample data object. This is automatically set when the sample is created and is based on the sample definition.

- getDefaultValue() - Object
  Returns the default value based on the attribute this sample data object is associated with. This is automatically set when the sample is created and is based on the sample definition.

- getMinValue() - Object
  Returns the minimum value based on the attribute this sample data object is associated with. This is automatically set when the sample is created and is based on the sample definition.

- getMaxValue() - Object
  Returns the maximum value based on the attribute this sample data object is associated with. This is automatically set when the sample is created and is based on the sample definition.

- getAttrValue() - Object
  Returns the data value for this sample data object.

- getAttrValueAsString() - String
  Returns the value for this sample data object as a string.

- setAttrValue(Object attrValue)
  Sets the value for this sample data object. If the attrValue parameter is not the correct data type, an attempt to convert it to the correct data type is performed before it is set.

- isValueValid() - boolean
  Returns true if the value of this sample data object has been set and is between minimum and maximum values.

- isModified() - boolean
  Returns true if the value of this sample data object has been modified.

3.6.2.3 Attribute Data Type

The attribute data type object contains the available data types of a sample attribute.

Available data types:

INTEGER

© Inductive Automation

Attribute can contain positive or negative numeric values with no fractions. It has a minimum value of -2,147,483,648 and a maximum value of 2,147,483,647 (inclusive).

REAL
Attribute can contain double-precision 64-bit IEEE 754 floating point values.

BOOLEAN
Attribute can contain a true or false value.

INSPECTED_COUNT
Attribute can contain a counting number (1, 2, 3, 4, …) and represents a number of items inspected for a attribute samples. This attribute data type is recognized and required by the p, np, c and u control charts.

NONCONFORMING_COUNT
Attribute can contain a counting number (1, 2, 3, 4, …) and represents a number of nonconforming items (defective items) for a attribute samples. This attribute data type is recognized and required by the p and np control charts.

NONCONFORMITY_COUNT
Attribute can contain a counting number (1, 2, 3, 4, …) and represents the number of nonconformities items that have (deformities) for a attribute samples. This attribute data type is recognized and required by the c and u control charts.

properties:

- getText() - String
  Returns the user friendly localized text for the attribute data type.

- intToType(int ordinal) - AttributeDataType
  Returns the attribute data type object for the ordinal value specified.

- dataTypeToType(DataType dataType) - AttributeDataType
  Returns the attribute data type object for the Ignition data type specified. For more information about DataType, see the Ignition documentation.

- getJavaType() - Class
  Returns the java data type.

- isNumeric() - boolean
  Returns true if the attribute data type handles numbers.

- isLogical() - boolean
  Returns true if the attribute data type is boolean.

- convert(Object attrValue) - Object
  Returns value in the true java data type for the type of data this attribute data type represents.

3.6.2.4 Sample Additional Factor

The sample additional factor object holds all of the information associated with one sample.

properties:

- getName() - String

Returns the name of this additional factor.

- getDataType() - DataType
  Returns the data type of this additional factor. See DataType in the Ignition documentation for more information.

- getValue() - Object
  Returns the value for this additional factor.

- updateValue(Object value, Date recordDateTime)
  Updates the value and the required date and time that is being record for this additional factor.

- getRecordDateTime() - Date
  Returns the date and time the value was recorded for this additional factor.

- isModified() - boolean
  Returns true if this additional factor has been modified.

3.6.2.5 Sample Definition

The sample definition object holds all of the information defining a type of sample. A sample definition specifies the attributes to collect, the locations where sample data is collected from, the control limits that apply, and the signals that apply.
When samples are created, they are based on a sample definition. And when sample measurement values are recorded, the sample definition is used to validate the measurement values. Other operations also refer back to the sample definition such as automatic scheduling of samples, auto evaluation of signals, etc.

properties:

- getDefUUID() - String
  Returns the UUID assigned to this sample definition. A UUID is a universally unique identifier that, once assigned to a sample definition, will never change. It is automatically generated when a sample definition is created and is unique in that no two samples definitions will have the same UUID.

- getName() - String
  Returns the name of this sample definition.

- setName(String name)
  Sets the name used for this sample definition. It is recommended that once samples have been created using this name, it should not be changed.

- getDescription() - String
  Returns the description of this sample definition.

- setDescription() - String
  Sets the description of this sample definition.

- getEnabled() - boolean
  Returns true if sample definition is enabled.

- setEnabled(boolean enabled)

© Inductive Automation

Sets sample definition enabled state.

- getMeasurementCount() - int
  Returns the number of measurements that this sample definition is configured for.

- setMeasurementCount(int measurementCount)
  Sets this number of measurement to be used when creating samples based on the sample definition.

- getIntervalType() - String
  Returns the default interval type for automatically scheduled samples based on this sample definition. Allowed locations that belong to this sample definition are initialized with this default interval type. The return value must match those configured on the Quality tab for the enterprise in the Sample Interval list.

- setIntervalType(String intervalType)
  Sets the default interval type for automatically scheduled samples. Allowed locations that belong to this sample definition are initialized with this default interval type. The return value must match those configured on the Quality tab for the enterprise in the Sample Interval list.

- getInterval() - double
  Returns the default interval for automatically scheduled samples based on this sample definition. Allowed locations that belong to this sample definition are initialized with this default interval. The units are defined by the Interval type defined for this sample definition.

- setInterval(double interval)
  Sets the default interval for automatically scheduled samples. Allowed locations that belong to this sample definition are initialized with this default interval. The units are defined by the Interval type defined for this sample definition.

- getAutoApprove() - boolean
  Returns the default auto approve samples setting for this sample definition. Allowed locations that belong to this sample definition are initialized with this default setting.

- setAutoApprove(boolean autoApprove)
  Sets the default auto approve samples setting for this sample definition. Allowed locations that belong to this sample definition are initialized with this default setting.

- getComingDueMin() - double
  Returns the default coming due minutes setting for this sample definition. Allowed locations that belong to this sample definition are initialized with this default setting. The value represents the number of minutes required before a sample is due until the sample is considered coming due. For automatically scheduled samples, they are created prior to actual due time by the number of minutes of this setting.

- setComingDueMin(double comingDueMin)
  Sets the default coming due minutes setting for this sample definition. Allowed locations that belong to this sample definition are initialized with this default setting. The value represents the number of minutes required before a sample is due until the sample is considered coming due. For automatically scheduled samples, they are created prior to actual due time by the number of minutes of this setting.

© Inductive Automation

- getOverdueMin() - double
  Returns the default overdue minutes setting for this sample definition. Allowed locations that belong to this sample definition are initialized with this default setting. The value represents the number of minutes required after a sample is due until the sample is considered overdue.

- setOverdueMin(double overdueMinutes)
  Sets the default overdue minutes setting for this sample definition. Allowed locations that belong to this sample definition are initialized with this default setting. The value represents the number of minutes required after a sample is due until the sample is considered overdue.

- isModified()
  Returns true if this sample definition has been modified.

- isNew()
  Returns true if this sample definition is new.

attribute properties:

- addAttribute(SampleDefinitionAttribute attribute) - String
  Adds a new attribute defined in the attribute parameter. Any error messages are returned, otherwise an empty string is returned.

- removeAttribute(SampleDefinitionAttribute attribute) - String
  Removes the attribute defined in the attribute parameter. Instead of attributes being permanently removed, thier enabled flag is set to false. Any error messages are returned, otherwise an empty string is returned.

- removeAttribute(int index) - String
  Removes the attribute defined in the index parameter. Instead of attributes being permanently removed, thier enabled flag is set to false. Any error messages are returned, otherwise an empty string is returned.

- removeAttribute(String name) - String
  Removes the attribute defined in the name parameter. Instead of attributes being permanently removed, thier enabled flag is set to false. Any error messages are returned, otherwise an empty string is returned.

- clearAttributes()
  All attributes contained in this sample definition are removed. Instead of the attributes being permanently removed, thier enabled flag is set to false.

- getAttribute(String name) - SampleDefinitionAttribute
  Returns the attribute with the same name as the name parameter.

- getAllAttribute() - List<SampleDefinitionAttribute>
  Returns a list of all attributes associated with this sample definition. This function will return enabled and disabled attributes.

- getEnabledAttributes() - List<SampleDefinitionAttribute>
  Returns a list of all attributes associated with this sample definition. This function will return only enabled attributes.

© Inductive Automation

- attributeExists(SampleDefinitionAttribute attribute) - boolean
  Returns true if the attribute specified in the parameter already exists for this sample definition. True will also be returned for disabled attributes.

allowed location properties:

- addAllowedLocation(SampleDefinitionLocation location) - String
  Adds a new allowed location defined in the location parameter. By adding an allowed location to this sample definition, this type of sample will appear as a option for the location and the real time location will be saved along with associated samples. For example, shift, product code, ref No and additional factor information is saved along with the sample data. Any error messages are returned, otherwise an empty string is returned.

- removeAllowedLocation(SampleDefinitionLocation location) - String
  Removes the allowed location defined in the location parameter. Allowed locations are permanently removed but can be added back. SPC data is not lost and will appear in the control charts and analysis. Any error messages are returned, otherwise an empty string is returned.

- removeAllowedLocation(int index) - String
  Removes the allowed location defined in the index parameter. Allowed locations are permanently removed but can be added back. SPC data is not lost and will appear in the control charts and analysis. Any error messages are returned, otherwise an empty string is returned.

- removeAllowedLocation(String locationName) - String
  Removes the allowed location defined in the locationName parameter. Allowed locations are permanently removed but can be added back. SPC data is not lost and will appear in the control charts and analysis. Any error messages are returned, otherwise an empty string is returned.

- clearAllowedLocations()
  All allowed locations contained in this sample definition are removed. Allowed locations are permanently removed but can be added back. SPC data is not lost and will appear in the control charts and analysis.

- getAllowedLocation(String name) - SampleDefinitionLocation
  Returns the allowed location with the same name as the name parameter.

- getAllAllowedLocations(boolean includeRemoved) - List<SampleDefinitionLocation>
  Returns a list of all allowed locations associated with this sample definition. If the includeRemoved parameter is true, the results will include removed allowed locations that have not been committed by saving the sample definition.

- allowedLocationExists(SampleDefinitionLocation location) - boolean
  Returns true if the allowed location specified in the parameter already exists for this sample definition. True will also be returned for allowed locations that have been removed, but not committed by saving the sample definition.

control limit properties:

- addControlLimit(SampleDefinitionControlLimit controlLimit) - String

© Inductive Automation

Adds a new control limit defined in the controlLimit parameter. By adding a control limit to this sample definition, it will show as an option in the control charts and may also be used when evaluating signals. The controlLimit parameter must be a valid control limit that appears in the enterprise production item. Any error messages are returned, otherwise an empty string is returned.

- removeControlLimit(SampleDefinitionControlLimit controlLimit) - String
  Removes the control limit defined in the controlLimit parameter. Any error messages are returned, otherwise an empty string is returned.

- removeControlLimit(int index) - String
  Removes the control limit defined in the index parameter. Any error messages are returned, otherwise an empty string is returned.

- removeControlLimit(String controlLimitName) - String
  Removes the control limit defined in the controlLimitName parameter. Any error messages are returned, otherwise an empty string is returned.

- clearControlLimits()
  All control limits contained in this sample definition are removed.

- getControlLimit(String name) -SampleDefinitionControlLimit
  Returns the control limit that has the same name as the name parameter.

- getAllControlLimits() - List<SampleDefinitionControlLimit>
  Returns all control limits that have been selected for this sample definition.

- getAllAllowedLocations(boolean includeRemoved) - List<SampleDefinitionLocation>
  Returns a list of all allowed locations associated with this sample definition. If the includeRemoved parameter is true the results will include removed allowed locations that have not been committed by saving the sample definition.

- controlLimitExists(SampleDefinitionControlLimit controlLimit) - boolean
  Returns true if the control limit specified in the parameter already exists for this sample definition.

signal properties:

- addSignal(SampleDefinitionSignal signal) - String
  Adds a new signal defined in the signal parameter. By adding a signal to this sample definition, it will show as an option in the control charts and may also be automatically evaluated. The signalparameter must be a valid signal that appears in the enterprise production item. Any error messages are returned, otherwise an empty string is returned.

- removeSignal(SampleDefinitionSignal signal) - String
  Removes the signal defined in the signal parameter. Any error messages are returned, otherwise an empty string is returned.

- removeSignal(int index) - String
  Removes the signal defined in the index parameter. Any error messages are returned, otherwise an empty string is returned.

© Inductive Automation

- removeSignal(String signalName) - String
  Removes the signal defined in the signalNameparameter. Any error messages are returned, otherwise an empty string is returned.

- clearSignals()
  All signals contained in this sample definition are removed.

- getSignal(String name) - SampleDefinitionSignal
  Returns the signal that has the same name as the name parameter.

- getAllSignals() - List<SampleDefinitionSignal>
  Returns all signals that have been selected for this sample definition.

- signalExists(SampleDefinitionSignal signal) - boolean
  Returns true if the signal specified in the parameter already exists for this sample definition.

3.6.2.6 Sample Definition Attribute

The sample definition attribute object holds all of the information defining an attribute used in samples. A sample definition attribute specifies the name, data type, default value and more for a single attribute that resides in a sample definition.

properties:

- getID() - int
  Returns the database created ID for this attribute.

- getParent() - SampleDefinition
  Returns the sample definition that this attribute is a child of.

- getNew() - SampleDefinitionAttribute
  Returns a new sample definition attribute instance.

- getName() - String
  Returns the name of this attribute.

- setName(String name)
  Sets the name used for this attribute. It is recommended that once samples have been created using this name, it should not be changed.

- getDescription() - String
  Returns the description of this attribute.

- setDescription() - String
  Sets the description of this attribute.

- getEnabled() - boolean
  Returns true if this attribute is enabled. If an attribute is disabled, it will not appear during sample entry. Based on the value of the included disabled attributes property on the SPC Selector component, disabled attributes will not show on the control charts.

- setEnabled(boolean enabled)

© Inductive Automation

Sets sample definition enabled state.

- getRequired() - boolean
  Returns true if this attribute is required while entering samples. If an attribute is required, a value must be entered before the sample will be saved.

- setRequired(boolean enabled)
  Sets this attribute required state. If an attribute is required, a value must be entered before the sample will be saved.

- getDatatype() - AttributeDataType
  Returns the attribute data type for this attribute. See Attribute Data Type for more information.

- setDatatype(AttributeDataType dataType)
  Sets this attribute's data type. See Attribute Data Type for more information.

- getFormat() - String
  Returns the format for this attribute. The format is used to verify formatting values on the control charts and that entered data is correctly formatted. See Attribute Data Type for more information.

- setFormat(String format)
  Sets this attribute's format. The format is used to verify formatting values on the control charts and that entered data is correctly formatted.
  Format strings consist of one or more of the characters shown in the table below. For example: the format string ##.0 will round to show one decimal place. Search java DecimalFormat for more information.

Symbol	Description
0	A digit. absent digits show as zero
#	A digit, zero shows as absent
.	Placeholder for decimal separator
,	Placeholder for grouping separator
E	Separates mantissa and exponent for exponential formats
;	Separates formats
-	Default negative prefix
%	Multiply by 100 and show as percentage
?	Multiply by 1000 and show as per mille
¤	Currency sign; replaced by currency symbol; if doubled, replaced by international currency symbol; if present in a pattern, the monetary decimal separator is used instead of the decimal separator
X	Any other characters can be used in the prefix or suffix
'	Used to quote special characters in a prefix or suffix

© Inductive Automation

Example format strings:

Value	Pattern	Output
123456.789	###,###.###	123,456.789
123456.789	###.##	123456.79
123.78	000000.000	000123.780
12345.67	$###,###.###	$12,345.67
12345.67	\u00A5###,###.###	¥12,345.67

- getDefaultValue() - Object
  Returns the default value for this attribute. If this optional default value exists, the sample's measurement values associated with this attribute are automatically set to this value when a sample is created.

- setDefaultValue(Object defaultValue)
  Sets the default value for this attribute. If this optional default value exists, the sample's measurement values associated with this attribute are automatically set to this value when a sample is created.

- getMinValue() - Object
  Returns the minimum value for this attribute. If this optional minimum value exists, the sample's measurement values associated with this attribute are required to be greater than or equal to this value.

- setMinValue(Object minValue)
  Sets the minimum value for this attribute. If this optional minimum value exists, the sample's measurement values associated with this attribute are required to be greater than or equal to this value.

- getMaxValue() - Object
  Sets the maximum value for this attribute. If this optional maximum value exists, the sample's measurement values associated with this attribute are required to be less than or equal to this value.

- setMaxValue(Object maxValue)
  Sets the maximum value for this attribute. If this optional maximum value exists, the sample's measurement values associated with this attribute are required to be less than or equal to this value.

- isModified()
  Returns true if this attribute definition has been modified.

- isNew()
  Returns true if this attribute definition is new.

© Inductive Automation

3.6.2.7 Sample Definition Location

The sample definition location object holds all of the information defining a location that samples are taken from. A sample definition location may specify the interval to schedule samples and various due time values. Be sure not to confuse a production location with the sample definition location object. The sample definition location object defines a production location that samples for a sample definition can be taken from.

When using the term Location within the SPC module, it refers to a virtual location where actual samples are taken. For example, if a sample bottle is taken from packaging line 1 and is tested in the lab for color, then the location is packaging line 1. In addition the the lab taking samples from this location, the operator can take samples to test labels. The tag property is used to define the ownership of who is responsible to take a sample.

properties:

- getID() - int
  Returns the database created ID for this sample definition location. Note, this is not the same as the production location ID.

- getNew(int locationID, String name) - SampleDefinitionLocation
  Returns a new sample definition location instance for the production location specified by the locationID parameter. The new instance name is specified by the name parameter.

- getLocationID() - int
  Returns the database created ID for the production location that this sample definition location is associated with. Note, this is the same as the production location ID.

- getParent() - SampleDefinition
  Returns the sample definition that this location is a child of.

- getName(String name)
  Returns the name of the production location associated with this sample definition location. This name also appears as the name for this sample definition location.

- getEnabled() - boolean
  Returns true if this sample definition location is enabled. If disabled, samples will not be automatically scheduled or appear in sample definition selection lists for the production location.

- setEnabled(boolean enabled)
  Sets sample definition location enabled state. If disabled, samples will not be automatically scheduled or appear in sample definition selection lists for the production location.

- getIntervalType() - String
  Returns the interval type for automatically scheduling samples for this location. The return value must match those configured on the Quality tab for the enterprise in the Sample Interval list.

- setIntervalType(String intervalType)
  Sets the interval type for automatically scheduling samples for this location. The intervalType value must match those configured on the Quality tab for the enterprise in the Sample Interval list.

© Inductive Automation

- getInterval() - double
  Returns the interval for automatically scheduling samples for this location. The units are defined by the Interval type defined for this sample definition.

- setInterval(double interval)
  Sets the interval for automatically scheduling samples for this location. The units are defined by the Interval type defined for this sample definition.

- getAutoApprove() - boolean
  Returns the auto approve setting for this location. If true, samples will be automatically approved when they are recorded. If false, they have to be manually approved.

- setAutoApprove(boolean autoApprove)
  Sets the auto approve setting for this location. If true, samples will be automatically approved when they are recorded. If false, they have to be manually approved.

- getComingDueMin() - double
  Returns the coming due minutes setting for this location. The value represents the number of minutes required before a sample is due until the sample is considered coming due. For automatically scheduled samples, they are created prior to actual due time by the number of minutes of this setting.

- setComingDueMin(double comingDueMin)
  Sets the coming due minutes setting for this location. The value represents the number of minutes required before a sample is due until the sample is considered coming due. For automatically scheduled samples, they are created prior to actual due time by the number of minutes of this setting.

- getOverdueMin() - double
  Returns the overdue minutes setting for this location. The value represents the number of minutes required before a sample is due until the sample is considered overdue.

- setOverdueMin(double overdueMinutes)
  Sets the overdue minutes setting for this location. The value represents the number of minutes required before a sample is due until the sample is considered overdue.

- getDuration() - double
  Returns the number of minutes needed to take a sample for this location.

- setDuration(double duration)
  Sets the number of minutes needed to take a sample for this location.

- getTag() - String
  Returns the tag setting for this location.

- setTag(String tag)
  Sets the tag setting for this location. The tag is used to assign ownership of who is responsible to take samples. For example, set to “Lab" if the lab is responsible or “Operator” if the operator is responsible.

© Inductive Automation

- isModified()
  Returns true if this sample definition has been modified.

- isNew()
  Returns true if this sample definition is new.

3.6.2.8 Sample Definition Control Limit

The sample definition control limit object holds all of the information defining a control limit that is applied to a sample definition. Be sure not to confuse a control limit defined in the Ignition designer with the sample definition control limit object. The sample definition control limit object connects a control limit defined in the Ignition designer with a sample definition.

Once a control limit is associated with a sample definition, it will appear as an option in the SPC Selector and can appear on control charts. It will also be included during automatic signal evaluations that require the control limit.

properties:

- getID() - int
  Returns the database created ID for this sample definition control limit.

- getParent() - SampleDefinition
  Returns the sample definition that this control limit is a child of.

- getName() - String
  Returns the name of this control limit as defined in the Ignition designer.

- setName(String name)
  Sets the name of this control limit as defined in the Ignition designer.

- getKind() - ControlLimitKindTypes
  Returns the the kind of control limit. There are different types of control limits and calculations for each type of chart category and this property makes this association between the two.

- setKind(ControlLimitKindTypes kind)
  Sets the the kind of control limit. There are different types of control limits and calculations for each type of chart category and this property makes this association between the two.

- setKind(int ordinal)
  Set the the kind of control limit based on a ControlLimitKindTypes ordinal value. There are different types of control limits and calculations for each type of chart category and this property makes this association between the two.

- getEnabled() - boolean
  Returns true if this sample definition control limit is enabled. If disabled, it will not show as an option on the control charts.

- setEnabled(boolean enabled)
  Sets this sample sample definition control limit enabled state. If disabled, it will not show as an option on the control charts.

- isModified()

© Inductive Automation

Returns true if this sample definition control limit has been modified.

- isNew()
  Returns true if this sample definition control limit is new.

3.6.2.9 Sample Definition Signal

The sample definition signal object holds all of the information defining a signal that will be applied to a sample definition. Be sure not to confuse a signal defined in the Ignition designer with the sample definition signal object. The sample definition signal object connects a signal defined in the Ignition designer with a sample definition.

Once a signal is associated with a sample definition, it will appear as an option in the SPC Selector and can appear on control charts. It will also be included during automatic signal evaluations.

properties:

- getID() - int
  Returns the database created ID for this sample definition signal.

- getParent() - SampleDefinition
  Returns the sample definition that this signal is a child of.

- getName() - String
  Returns the name of this signal as defined in the Ignition designer.

- setName(String name)
  Sets the name of this signal as defined in the Ignition designer.

- getKind() - SignalKindTypes
  Returns the the kind of signal. There are different types of signals and calculations for each type of chart category and this property makes this association between the two.

- setKind(SignalKindTypes kind)
  Sets the the kind of signal. There are different types of signals and calculations for each type of chart category and this property makes this association between the two.

- setKind(int ordinal)
  Sets the the kind of signal based on a SignalKindTypesordinal value. There are different types of signals and calculations for each type of chart category and this property makes this association between the two.

- getEnabled() - boolean
  Returns true if this sample definition signal is enabled. If disabled, it will not show as an option on the control charts.

- setEnabled(boolean enabled)
  Sets this sample sample definition signal enabled state. If disabled, it will not show as an option on the control charts.

- isModified()
  Returns true if this sample definition signal has been modified.

© Inductive Automation

- isNew()
  Returns true if this sample definition signal is new.

3.6.2.10 Control Limit Kind Type

The control limit kind type object contains the available types of control limits. In all cases, the ending of the name specifies how it is used in control charts and automatic signal evaluation. An ending of _UCL is handled as a upper control limit, for _LCL it is handled as lower control limit and _OTHER is a general control limit.

Available data types:

XBAR_UCL
XBAR_LCL
XBAR_OTHER
Used for the XBar control chart.

RANGE_UCL
RANGE_LCL
RANGE_OTHER
Used for the Range control chart.

STDDEV_UCL
STDDEV_LCL
STDDEV_OTHER
Used for the s (standard deviation) control chart.

INDV_UCL
INDV_LCL
INDV_OTHER
Used for the Individual control chart.

MEDIAN_UCL
MEDIAN_LCL
MEDIAN_OTHER
Used for the Median control chart.

P_UCL
P_LCL
P_OTHER
Used for the p control chart.

NP_UCL
NP_LCL
NP_OTHER
Used for the np control chart.

C_UCL
C_LCL
C_OTHER
Used for the c control chart.

U_UCL

© Inductive Automation

U_LCL
U_OTHER
Used for the u control chart.

HISTOGRAM_UCL
HISTOGRAM_LCL
HISTOGRAM_OTHER
Used for the Histogram chart.

MOVING_RANGE_UCL
MOVING_RANGE_LCL
MOVING_RANGE_OTHER
Used for the MA (moving average) control chart.

properties:

- getText() - String
  Returns the user friendly localized text for the control limit kind.

- intToType(int ordinal) - ControlLimitKindTypes
  Returns the control limit kind type object for the ordinal value specified.

- getTypeFromName(String name) - ControlLimitKindTypes
  Returns the control limit kind type object for the name value specified

- getCategory() - SPCCategoryTypes
  Returns the category of chart. See SPC Category Types for more information.

3.6.2.11 SPC Category Types

The SPC category type defines the possible types of charts currently supported by the SPC module.

Available data types:

XBAR
RANGE
SBAR
INDIVIDUAL
MEDIAN
P
NP
U
C
HISTOGRAM
PARETO
MR

3.6.2.12 Signal Kind Types

The signal kind type object contains the available types that a signal can be.

Available data types:

XBAR
RANGE
SBAR
INDIVIDUAL

MEDIAN
P
NP
U
C
HISTOGRAM
PARETO
MR

properties:

- getText() - String
  Returns the user friendly localized text for the signal kind.

- intToType(int ordinal) - SignalKindTypes
  Returns the signal kind type object for the ordinal value specified.

- getTypeFromName(String name) - SignalKindTypes
  Returns the signal kind type object for the name value specified

- getCategory() - SPCCategoryTypes
  Returns the category of chart. See SPC Category Types for more information.

3.6.2.13 Control Limit Calculated Value

When using the calcControlLimitValue functions, the new calculated control limit value and any messages are returned in this object. Most control limits are a single value across all samples. The p and u chart control limits can have different values for each sample. In this case, the results are returned in a Dataset that is also in this object.

properties:

- getCalculatedValue() - double
  Returns the new single control limit value.

- getData() - Dataset
  Returns multiple control limit value that vary for each sample.

- hasMessage() - boolean
  True is returned if a message exists.

- getMessage() - String
  Message of why the control limit cannot be calculated.

3.6.3 Scripting Functions

3.6.3.1 sample.production

3.6.3.1.1 sample.production.utils

3.6.3.1.1.1 cancelLocationProductCode

system.production.utils.cancelLocationProductCode- Cancel Product Code

Description

Cancel the current product code for a production location. This is provided to handle no production being run at a production location.

Syntax

Client

```
cancelLocationProductCode(String locationPath)
```

Gateway

```
cancelLocationProductCode(String projectName, String locationPath)
```

Parameters

String locationPath - The full path of the location to set the product code.
String productCode - The new product code.
String refNo - Optional reference number.
String projectName - Name of the Ignition SPC project.

Returns

none

Scope

client, gateway

Example

```
#This is a sample gateway script to change a location's product code based on a
SQLTag value.
pc = system.tag.getTagValue('[Default]Quality/Test/ProductCodeTest')
if pc == '':
   system.production.utils.cancelLocationProductCode('QualityDemo',
   'QualityDemo\New Enterprise\New Site\Packaging\Line 1\Line 1 Quality')
else:
   system.production.utils.setLocationProductCode('QualityDemo', 'QualityDemo\New
   Enterprise\New Site\Packaging\Line 1\Line 1 Quality', pc, '')
```

3.6.3.1.1.2 setLocationProductCode

system.production.utils.setLocationProductCode- Set Product Code

Description

Set the product code and optional reference number for a production location. If the production location is already assigned a product code, then it will be canceled and the new product code will be set.

Note that this scripting function will not immediately change the product code. This is because if a production location is already assigned a product code, it requires two steps to change the current product code. The current product code must be canceled before the new product code is made active. This can be an issue if using this script function in the Before Sample Update Event. The product code will not be updated until after the sample is updated. Instead use get the sample from current event object and set the product code directly in the sample.

Syntax

Client

```
setLocationProductCode(String locationPath, String productCode, String refNo)
```

Gateway

```
setLocationProductCode(String projectName, String locationPath, String
productCode, String refNo)
```

Parameters

String locationPath - The full path of the location to set the product code.
String productCode - The new product code.
String refNo - Optional reference number.
String projectName - Name of the Ignition SPC project.

Returns

none

Scope

client, gateway

Example

```
#This is a sample gateway script to change a location's product code based on a
SQLTag value.
pc = system.tag.getTagValue('[Default]Quality/Test/ProductCodeTest')
if pc == '':
   system.production.utils.cancelLocationProductCode('QualityDemo',
   'QualityDemo\New Enterprise\New Site\Packaging\Line 1\Line 1 Quality')
else:
   system.production.utils.setLocationProductCode('QualityDemo', 'QualityDemo\New
   Enterprise\New Site\Packaging\Line 1\Line 1 Quality', pc, '')
```

3.6.3.2 sample.quality

3.6.3.2.1 sample.quality.definition

3.6.3.2.1.1 getNew

system.quality.definition.getNew

Description

Creates and returns a new instance of a SampleDefinition object.

Syntax

Client

```
system.quality.definition.getNew()
```

Gateway

```
system.quality.definition.getNew()
```

Parameters

none

Returns

SampleDefinition - new sample definition instance

Scope

client, gateway

3.6.3.2.1.2 getSampleDefinition

system.quality.definition.getSampleDefinition- by definition ID

Description

Returns a reference to the sample definition with a matching ID. The ID is generated by the database when the sample definition was first saved.

Syntax

Client

```
system.quality.definition.getSampleDefinition(int sampleDefID)
```

Gateway

```
system.quality.definition.getSampleDefinition(String projectName, int
sampleDefID)
```

Parameters

int sampleDefID - Database created ID for the sample definition.
String projectName - Name of the Ignition SPC project.

Returns

SampleDefinition - A reference to the matching sample definition

Scope

client, gateway

system.quality.definition.getSampleDefinition- by definition name

Description

Returns a reference to the sample definition with a matching name.

Syntax

Client

```
system.quality.definition.getSampleDefinition(String sampleDefName)
```

Gateway

```
system.quality.definition.getSampleDefinition(String projectName, String
sampleDefName)
```

Parameters

String sampleDefName - The name given to the sample definition when it was created.
String projectName - Name of the Ignition SPC project.

Returns

SampleDefinition - A reference to the matching sample definition

Scope

client, gateway

3.6.3.2.1.3 addSampleDefinition

system.quality.definition.addSampleDefinition

Description

Adds the sample definition passed in the parameter to the SPC system. After it has been added it will become available to record samples and for selection on the control charts. Attributes, locations, control limits and signals must be added to the sample definition prior to calling this function. See Sample Definition for more information.

Syntax

Client

```
system.quality.definition.addSampleDefinition(SampleDefinition
sampleDefinition)
```

Gateway

```
system.quality.definition.addSampleDefinition(String projectName,
SampleDefinition sampleDefinition)
```

Parameters

String sampleDefinition - New sample definition that previously was created in script.
String projectName - Name of the Ignition SPC project.

Returns

none

Scope

client, gateway

3.6.3.2.1.4 updateSampleDefinition

system.quality.definition.updateSampleDefinition

Description

Updates an existing sample definition passed in the parameter. After it has been updated, the changes will be reflected during recording samples and on the control charts.

Syntax

Client

```
system.quality.definition.updateSampleDefinition(SampleDefinition
sampleDefinition)
```

Gateway

```
system.quality.definition.updateSampleDefinition(String projectName,
SampleDefinition sampleDefinition)
```

Parameters

String sampleDefinition - Existing sample definition.
String projectName - Name of the Ignition SPC project.

Returns

none

Scope

client, gateway

3.6.3.2.2 sample.quality.sample.data

3.6.3.2.2.1 getNew ByDefUUID

system.quality.sample.data.getNewByDefUUID

Description

Creates and returns a new sample based on the sample definition that matches the defUUID parameter. The newly created sample will also be initialized for the location specified by the locationPath parameter.

Syntax

Client

```
system.quality.sample.data.getNewByDefUUID(String defUUID, String locationPath)
```

Gateway

```
system.quality.sample.data.getNewByDefUUID(String projectName, String defUUID,
String locationPath)
```

Parameters

String defUUID - Existing sample definition UUID to base this sample on.

String locationPath - A valid path to a location.
String projectName - Name of the Ignition SPC project.

Returns
Sample - A reference to the newly created sample.

Scope
client, gateway

Example

```
locationPath = event.source.parent.LocationPath
defUUID = event.newValue
sample = system.quality.sample.data.getNewByDefUUID(defUUID , locationPath)
```

3.6.3.2.2.2 getNew ByName

system.quality.sample.data.getNewByName

Description

Creates and returns a new sample based on the sample definition that matches the definitionName parameter. The newly created sample will also be initialized for the location specified by the locationPath parameter.

Syntax

Client

```
system.quality.sample.data.getNewByDefName(String defName, String locationPath)
```

Gateway

```
system.quality.sample.data.getNewByDefName(String projectName, String defName,
String locationPath)
```

Parameters
String defName - Existing sample definition name to base this sample on.
String locationPath - A valid path to a location.
String projectName - Name of the Ignition SPC project.

Returns
Sample - A reference to the newly created sample.

Scope
client, gateway

Example

```
locationPath = event.source.parent.LocationPath
sampleDefName = event.newValue
sample = system.quality.sample.data.getNewByDefName(sampleDefName, locationPath)
```

3.6.3.2.2.3 getCreateSampleByDefUUID

system.quality.sample.data.getCreateSampleByDefUUID

Description

Return a sample that matches the sampleUUID parameter. If not found, create and return a new sample based on the sample definition that matches the definitionUUID parameter. The newly created sample will also be initialized for the location specified by the locationPath parameter.

Syntax

Client

```
system.quality.sample.data.getCreateSampleByDefUUID(String sampleUUID, String
defUUID, String locationPath)
```

Gateway

```
system.quality.sample.data.getCreateSampleByDefUUID(String projectName, String
sampleUUID, String defUUID, String locationPath)
```

Parameters

String sampleUUID - Sample UUID to lookup.
String defUUID - Existing sample definition UUID to base the new sample on.
String locationPath - A valid path to a location to base the new sample for.
String projectName - Name of the Ignition SPC project.

Returns

Sample - A reference to the existing sample or the newly created sample.

Scope

client, gateway

Example

```
sampleUUID = system.gui.getParentWindow(event).getComponentForPath('Root
Container').SampleUUID
locationPath = system.gui.getParentWindow(event).getComponentForPath('Root
Container').LocationPath
#This will return a sample for the sampleUUID. If the sampleUUID is blank, it will
return a new sample
sample = system.quality.sample.data.getCreateSampleByName(sampleUUID, sampleDef.
getDefUUID(), locationPath)
```

3.6.3.2.2.4 getCreateSampleByName

system.quality.sample.data.getCreateSampleByName

Description

Return a sample that matches the sampleUUID parameter. If not found, create and return a new sample based on the sample definition that matches the definitionName parameter. The newly created sample will also be initialized for the location specified by the locationPath parameter.

Syntax

Client

```
system.quality.sample.data.getCreateSampleByName(String sampleUUID, String
defName, String locationPath)
```

Gateway

```
system.quality.sample.data.getCreateSampleByName(String projectName, String
sampleUUID, String defName, String locationPath)
```

Parameters

String sampleUUID - Sample UUID to lookup.
String defName - Existing sample definition name to base the new sample on.
String locationPath - A valid path to a location to base the new sample for.
String projectName - Name of the Ignition SPC project.

Returns

Sample - A reference to the existing sample or the newly created sample.

Scope

client, gateway

Example

```
sampleUUID = system.gui.getParentWindow(event).getComponentForPath('Root
Container').SampleUUID
locationPath = system.gui.getParentWindow(event).getComponentForPath('Root
Container').LocationPath
#This will return a sample for the sampleUUID. If the sampleUUID is blank, it
will return a new sample
sample = system.quality.sample.data.getCreateSampleByName(sampleUUID,
'Viscosity', locationPath)
```

3.6.3.2.2.5 getSample

system.quality.sample.data.getSample

Description

Return a sample that matches the sampleUUID parameter.

Syntax

Client

```
system.quality.sample.data.getSample(String sampleUUID)
```

Gateway

```
system.quality.sample.data.getSample(String projectName, String sampleUUID)
```

Parameters

String sampleUUID - Sample UUID to lookup.
String projectName - Name of the Ignition SPC project.

Returns

Sample - A reference to the existing sample.

Scope

client, gateway

Example

```
sampleUUID = system.gui.getParentWindow(event).getComponentForPath('Root
Container').SampleUUID
sample = system.quality.sample.data.getSample(sampleUUID)
```

3.6.3.2.2.6 updateSample

system.quality.sample.data.updateSample

Description

Update an existing or new sample. If the valuesRecorded parameter is true, current shift, product code and additional factor information will be recorded along with the measurement values. Because sample are scheduled, they can be created and updated with no measurement values. This allow for coming due, due and overdue functionality to be tracked.

Syntax

Client

```
system.quality.sample.data.updateSample(String locationPath, Sample sample,
Boolean valuesRecorded)
```

Gateway

```
system.quality.sample.data.updateSample(String projectName, String
locationPath, Sample sample, Boolean valuesRecorded)
```

Parameters

String locationPath - A valid path to a location to record this sample for.
Sample sample - Sample to update.

Boolean valuesRecorded - If true, record the values along with the other sample information.
String projectName - Name of the Ignition SPC project.

Returns
none

Scope
client, gateway

Example

```
system.quality.sample.data.updateSample('QualityDemo\New Enterprise\New
Site\Packaging\Line 1\Line 1 Quality', currentSample, Boolean 1)
```

3.6.3.2.2.7 approveSample

system.quality.sample.data.approveSample

Description

Approve an existing sample. If the associated sample definition for the specified sample is not set for auto approval, it will have to approved. This can be done using various methods of which this is one of them.

Syntax

Client

```
system.quality.sample.data.approveSample(String sampleUUID, String approvedBy)
```

Gateway

```
system.quality.sample.data.approveSample(String projectName, String
sampleUUID, String approvedBy)
```

Parameters

String sampleUUID - The UUID to an existing sample to approve.
String approvedBy - The name of the person who is approving the sample.
String projectName - Name of the Ignition SPC project.

Returns
none

Scope
client, gateway

Example

```
system.quality.sample.data.approveSample(currentSample.getSampleUUID, system.
security.getUsername())
```

3.6.3.2.2.8 unapproveSample

system.quality.sample.data.unapproveSample

Description

Unapprove a previously approved sample. When a sample is unapproved it will not be shown in the control charts or included in the data during automatic signal evaluation.

Syntax

Client

```
system.quality.sample.data.unapproveSample(String sampleUUID)
```

Gateway

```
system.quality.sample.data.unapproveSample(String projectName, String
sampleUUID)
```

Parameters

String sampleUUID - The UUID to an existing sample to approve.
String projectName - Name of the Ignition SPC project.

Returns
none

Scope
client, gateway

Example
```
system.quality.sample.data.unapproveSample(currentSample.getSampleUUID)
```

3.6.3.2.2.9 removeSample

system.quality.sample.data.removeSample

Description
Remove a single sample. This function should be used with caution because it permanently removes the data from the database. A sample can be removed at any point in its life cycle. Meaning it can be removed after it has been scheduled but before measurements are recorded and after measurements have been recorded.

Syntax
Client
```
system.quality.sample.data.removeSample(String sampleUUID)
```
Gateway
```
system.quality.sample.data.removeSample(String projectName, String sampleUUID)
```
Parameters
String sampleUUID - The UUID to an existing sample to approve.
String projectName - Name of the Ignition SPC project.

Returns
none

Scope
client, gateway

Example
```
system.quality.sample.data.removeSample(event.getSampleUUID())
```

3.6.3.2.3 sample.quality.spc.controllimit

Enter topic text here.

3.6.3.2.3.1 setControlLimitValue

system.quality.spc.controllimit.setControlLimitValue

Description
Control limits normally are set using the control charts components and when the process is determined to be stable. In cases where additional flexibility is required, this scripting function is provided. New control limit values for a specified location, sample definition (test), attribute and control limit can be set by calling this function

Syntax
Client
```
setControlLimitValue(String locationPath, SampleDefinition definition, String
attributeName, String limitName, double value)
```

Gateway

```
setControlLimitValue(String projectName, String locationPath, SampleDefinition
definition, String attributeName, String limitName, double value)
```

Parameters

String locationPath - The full path of the location to set the control limit. Optionally, it can be left blank to set the default control limit value that is not tied to any location.
SampleDefinition definition - Sample definition to the control limit for.
String attributeName - Name of the attribute within the definition to set the control limit for.
String limitName - Name of the control limit to set.
double value - New control limit value.
String projectName - Name of the Ignition SPC project.

Returns

none

Scope

client, gateway

Example

```
#This is a sample client script to change a control limit to a fixed value.
system.quality.spc.controllimit.setControlLimitValue('New Enterprise\New
Site\Packaging\Line 1\Line 1 Quality', sampleDef, 'Weight', 'Individual LCL',
100.0)
```

3.6.3.2.3.2 calcControlLimitValue

system.quality.spc.controllimit.calcControlLimitValue

Description

Control limits normally are calculated using the control charts components and when the process is determined to be stable. In cases where additional flexibility is required, this scripting function is provided to calculate control limits from data provided in the parameters. Control limit values for a specified location, sample definition (test), attribute and control limit can be calculated by calling this function. The control limit will be calculated using the the control limit configured in the designer and the data specified in the parameters. To set the actual control limit value use the setControlLimitValue function with the result from this function.

Syntax

Client

```
calcControlLimitValue(String locationPath, SampleDefinition definition, String
attributeName, String limitName, Dataset data)
```

Gateway

```
calcControlLimitValue(String projectName, String locationPath,
SampleDefinition definition, String attributeName, String limitName, Dataset
data)
```

Parameters

String locationPath - The full path of the location to set the control limit. Optionally, it can be left blank to set the default control limit value that is not tied to any location.
SampleDefinition definition - Sample definition to the control limit for.
String attributeName - Name of the attribute within the definition to set the control limit for.
String limitName - Name of the control limit to set.
Dataset data - A dataset containing SPC results to calculate the control limit from.
String projectName - Name of the Ignition SPC project.

Returns

ControlLimitCalculatedValue - A reference to the results containing the calculated control limit and any messages. See Control Limit Calculated Value for more information.

Scope
client, gateway

system.quality.spc.controllimit.calcControlLimitValue

Description
Control limits normally are calculated using the control charts components and when the process is determined to be stable. In cases where additional flexibility is required, this scripting function is provided to calculate control limits from a date range provided in the parameters. This function will collect data within the from and end dates specified in the parameters. It calculates a control limit for the specified location, sample definition (test), attribute and control limit. The control limit will be calculated using the the control limit configured in the designer. To set the actual control limit value use the setControlLimitValue function with the result from this function.

Syntax
Client
```
calcControlLimitValue(String locationPath, SampleDefinition definition, String attributeName, String limitName, Date from, Date to)
```
Gateway
```
calcControlLimitValue(String projectName, String locationPath, SampleDefinition definition, String attributeName, String limitName, Date from, Date to)
```
Parameters
String locationPath - The full path of the location to set the control limit. Optionally, it can be left blank to set the default control limit value that is not tied to any location.
SampleDefinition definition - Sample definition to the control limit for.
String attributeName - Name of the attribute within the definition to set the control limit for.
String limitName - Name of the control limit to set.
Date from - Calculate the control with data starting with this date.
Date to - Calculate the control with data ending with this date.
String projectName - Name of the Ignition SPC project.
Returns
ControlLimitCalculatedValue - A reference to the results containing the calculated control limit and any messages. See Control Limit Calculated Value for more information.
Scope
client, gateway

Example
```
#This is a sample client script to change a control limit to a fixed value.
#Define the starting date to calculate the control limit
from java.util import Calendar
fromDate = Calendar.getInstance();
fromDate.add(Calendar.DAY_OF_MONTH, -1)

#Define the endingdate to calculate the control limit
toDate = Calendar.getInstance();
#Get the sample definition based on its name
sampleDef = system.quality.definition.getSampleDefinition('SQLTag-Line 1 Checkweigher')

#Calculate the new control limit value
result = system.quality.spc.controllimit.calcControlLimitValue('New
```

```
Enterprise\New Site\Packaging\Line 1\Line 1 Quality', sampleDef, 'Weight',
'Individual LCL', fromDate.getTime(), toDate.getTime())

#Check the results to make sure there are no messages
if result != None and result.hasMessage() == 0:
      #Set the actual control limit to the new calculated value
      system.quality.spc.controllimit.setControlLimitValue('New Enterprise\New
      Site\Packaging\Line 1\Line 1 Quality', sampleDef, 'Weight', 'Individual
      LCL', result.getCalculatedValue())
```

3.7 Analysis Providers

Analysis providers determine which information will be viewed on a graph or pie chart. Based on which Analysis Provider is selected, some filter, compare by, and data point options may or may not be visible. This section covers only the Quality Analysis Provider that is available with the SPC module.

Quality Analysis Provider

3.7.1 Quality

Description

The Quality Analysis Provider is used to query SPC information that is beyond what can be shown on control charts. For example, to determine the number of samples taken by user or the number of times a process was out of control over the last month cannot easily be shown in a control chart.

Provider Name

Quality

Filters

These are the filters that are available in the SPC Module. However, in addition to these filters, additional factors may be available if they are string data type. All additional factors start with "Factor:". For example, "Factor:Operator". A filter will allow the user to see all of the data points in the analysis provider as it pertains to a specific *area*, shift, etc. For more information on filters, see the Filter By paragraph in the Analysis Screen section.

Area

Attribute Name

Definition Name

Enterprise

Include

Line

Location

Product Code

Reference Number

Sample Note

Shift

Shift Sync

Site

Tag

Compare By

These are the comparisons that are available in the SPC Module. However, in addition to these comparisons, additional factors may be available if they are string data type. All additional factors start with "Factor:". For example, "Factor:Operator". A comparison allows one data point to be compared between all *areas*, days, etc. For more information on comparisons, see the Compare By paragraph in the Analysis Screen section.

Approved By

Area

Attribute Name

Day

Definition Name

Enterprise

Line

Location

Month

Note Entered By

Product Code

Reference Number

Sample Entered At

Sample Taken By

Shift

Site

Tag

Week

Year

Data Points

These are the data points that are available in the SPC Module. However, in addition to these comparisons, additional factors may be available if they are string data type. All additional factors start with "Factor:". For example, "Factor:Operator". Data points are the different values that will be presented or compared on a graph or chart. For more information on data points, see the Data Point paragraph in the Analysis Screen section.

© Inductive Automation

Approved At
Approved By
Approved Count
Area
Attribute Name
Attribute Note
Day
Definition Name
Enterprise
Line
Location
Month
Note Entered By
Product Code
Reference Number
Sample Count
Sample Entered At
Sample Note
Sample Taken At
Sample Taken By
Scheduled Finish
Schedule Start
Shift
Site
Tag
Week
Year

© Inductive Automation

Recipe / Changeover

Part IV

inductive automation
800.266.7798
www.inductiveautomation.com

4 Recipe / Changeover

4.1 Introduction

The Recipe module extends Ignition to manage and monitor recipes. It is ideal for quickly and accurately changing machine, process or system recipes. Powerful master recipe and sub-recipe management, recipe security, change log tracking, variance tracking and more empower you to improve efficiency and quality, and take more control of your manufacturing facility.

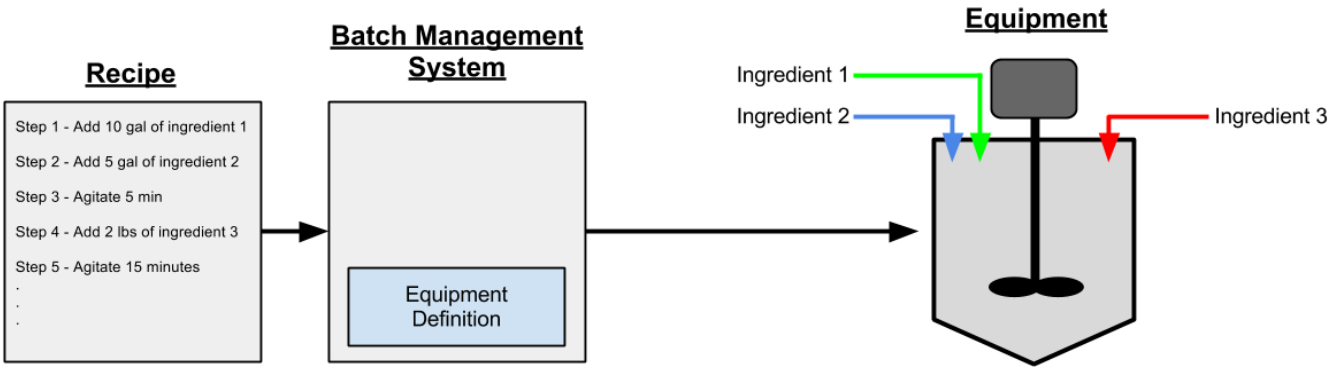
4.1.1 Recipe Types

Batch Recipes

We commonly think of recipes as making a batch of product. An analogy to this is a batch of cookies where many ingredients are added in sequence along with mixing. It is important to understand that a batch system is different from a recipe. It is true that batch systems use recipes, but a batch system has equipment definitions that are combined with the recipe to control the machinery to make a batch of product.

Batch Management Systems handle many other functions including inventory checks before starting a batch, alarm detection, machine control and more.

The Recipe / Changeover Module does not do the functions of a Batch Management System. This being said, you can add multiple steps as child recipes to a master recipe and then step or sequence through the steps. The sequencing through the steps must be done in Ignition script or the PLC.


Recipe
Step 1 - Add 10 gal of ingredient 1
Step 2 - Add 5 gal of ingredient 2
Step 3 - Agitate 5 min
Step 4 - Add 2 lbs of ingredient 3
Step 5 - Agitate 15 minutes
Batch Management System
Equipment Definition
Equipment
Ingredient 1
Ingredient 2
Ingredient 3

Batch System

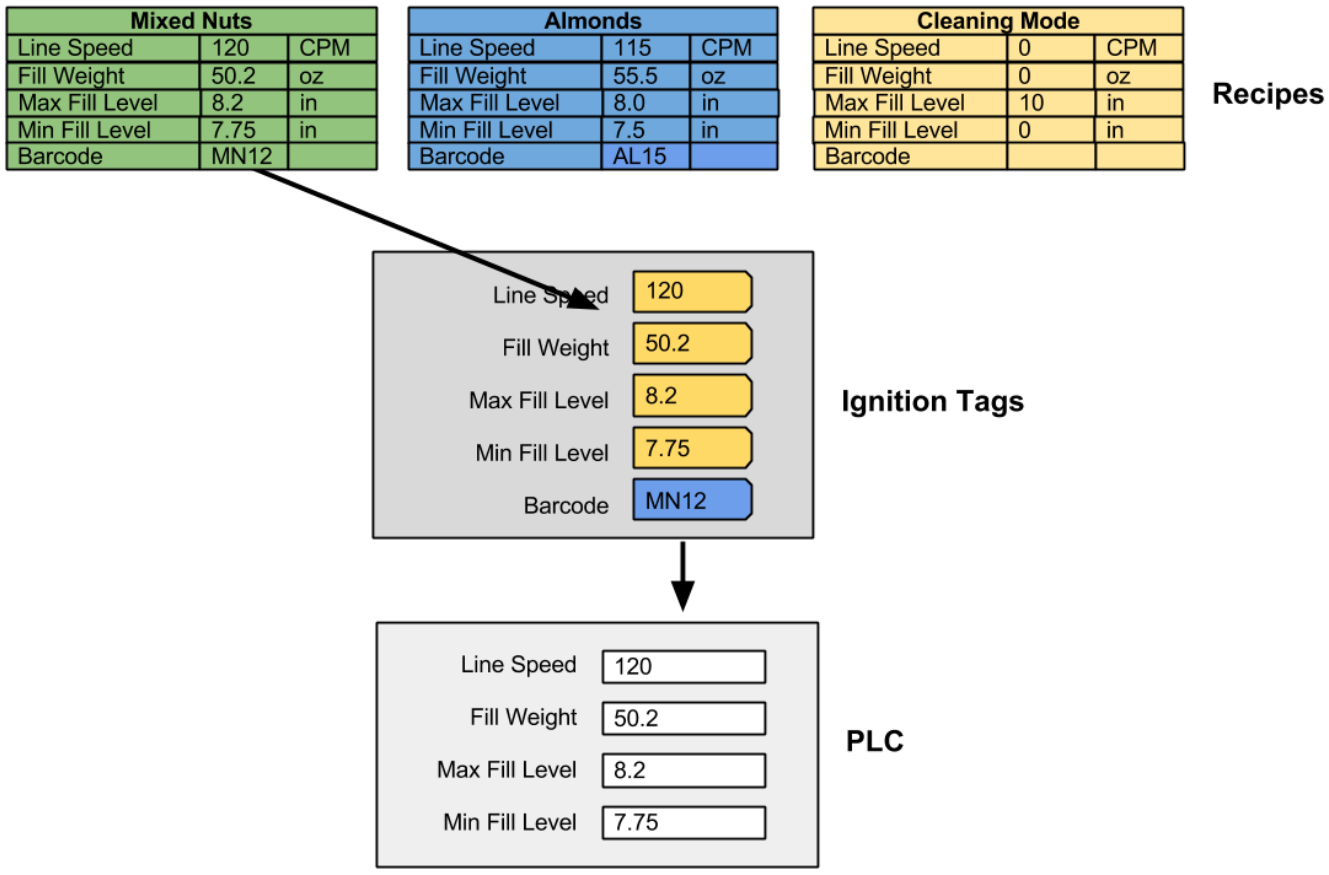
Machine Recipes

Machine recipes are used for setup equipment to run a given product or to put it in a given mode. If a machine can run 20 different products and each product has different settings, then the need to manage recipes is essential. Commonly, machines have some sort of operator interface that will allow the operator to change settings and in some cases, have a very basic recipe system. This can work okay for a single machine but with production lines where there are several machines, it becomes more of a task to go to each machine and make sure it is setup to run the next product on the schedule. This requires time and is prone to mistakes during changeover between products.

© Inductive Automation

When a recipe (product code) is selected for a machine, the recipe values are written to Ignition tags, some or all of the Ignition tags can be mapped to memory locations in a PLC. In the image below, all the recipe values except for the Barcode are mapped to a PLC through OPC. The Barcode recipe value is just mapped to an Ignition memory tag and can be displayed on a screen for the operator to verify the barcode number or it can be sent to a printer through serial or TCP/IP.

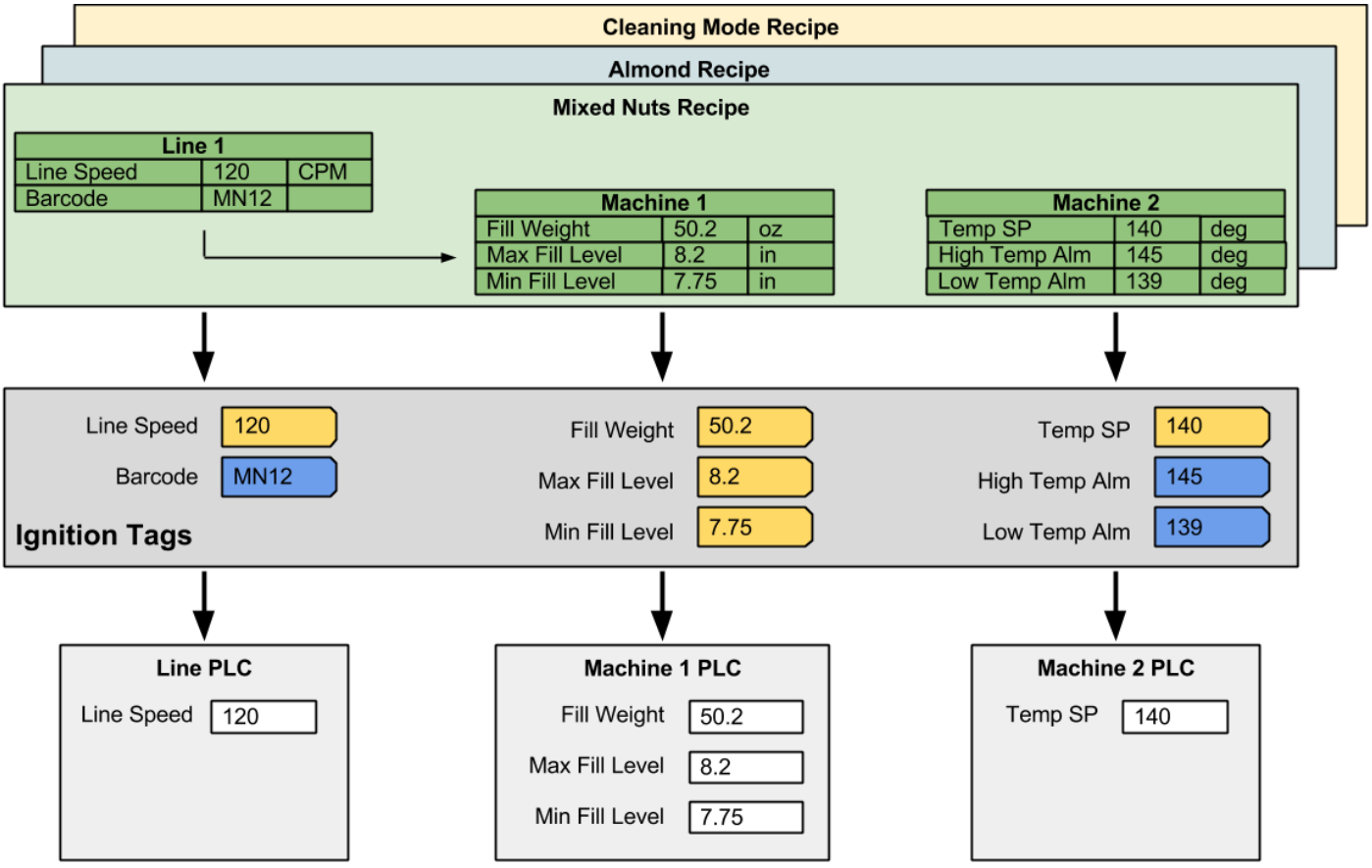
If the Almonds recipe is selected, the recipe value will be written to the Ignition tags. If the Ignition tags are tied to PLC memory addresses, they will end up in the PLC and the machine will be ready to run almonds.



Single Machine Recipe Value Tag Mapping

When a production line contains production cells, cell groups or locations as children, the recipes can be managed, selected and reported on by the production line. The image below depicts this where Line 1 has two machines as children. These children can be production cells, cell groups or locations. If the machines are being tracked with the OEE Downtime Module, then the existing production cells or cell groups should be reused. However, if the machines are not being tracked with the OEE Downtime Module, then locations can be used. See Production Model for more information.

© Inductive Automation



Multiple Machine Recipe Value Tag Mapping

This is a brief overview of the types of recipes that are commonly used in manufacturing. There is a lot more functionality such as scaling, variance monitoring, change logs, master recipes, sub recipes, reporting capabilities, etc. that comes along with the Recipe / Changeover Module that is covered in the following sections of this manual.

4.1.2 Production Model

To start out, it is important to define what the production model is, which is heavily referred to when dealing with recipe values. Recipe values are defined by machine, or in some cases a virtual location. Once recipe values are defined for a machine, they can be added to recipes. After which, the recipe can be selected for the machine.

A production model defines your manufacturing or process in tree view form. It provides an organized way to easily configure, control and analyze your facility.

Production Model Tree

© Inductive Automation

Enterprise

The *enterprise* is the highest level of the production model and typically represents a manufacturing company. A company may have one or more production facilities (sites).

Site

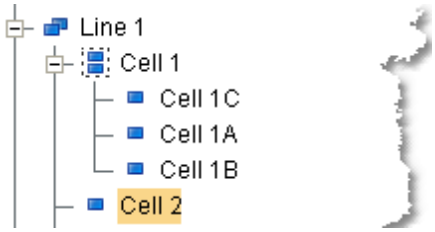
A *site* is a geographical production location and is part of an *enterprise*.

Area

An *area* is a physical or logical grouping of production *lines*.

Line

A *line* is a collection of one or more cells and/or cell groups and/or locations that run a single product at any given time.



Cell Group Tree

Location

A location is the space where a sample is collected, product is tracked through or recipes are selected for. This can be placed under an area or a line.

Cell Group

A *cell group* contains two or more *cells*. Typically, these *cells* occur at the same time in the sequence of the *line* instead of one after another, causing the *cell group* to act as a single sub process or step within the production.

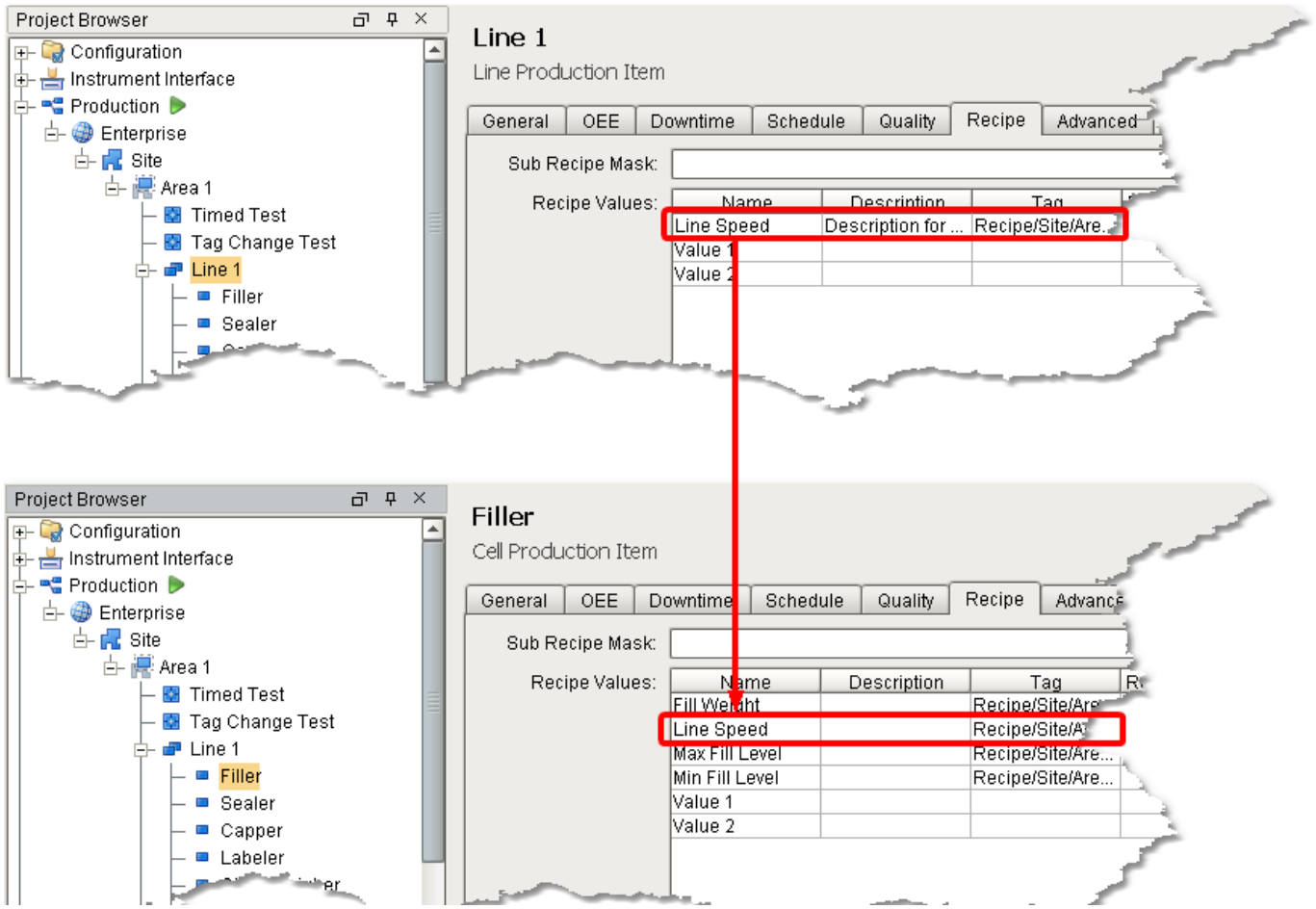
Cell

The *cell* is a single machine, sub process or step required in the manufacturing of a product. Cells are used for tracking OEE and downtime but can also have recipe values added to them and can be used for track and trace. The product may be a hard product such as used in packaging, liquid, powder, etc. Packaging machines are a common example, but a cell applies to processes also.

Recipe Value Propagation

Recipe values that are added to a production item are propagated down to the child production items. For example, if LineSpeed recipe value is added to a production line, then all cells, cell groups and locations that are children of the production line, will also have the LineSpeed recipe value. The Ignition tag associated with the recipe value is not propagated to the child recipe value.

© Inductive Automation



Recipe Value Propagation

Only the recipe values that have Ignition tags assigned to them will appear in the recipe editor. So, if a propagated recipe value is not relevant to the child production item, the recipe value Tag property can be left blank.

4.1.3 Default Values

Default Values

When a new recipe is created, it is initialized with a default value. If the recipe value is assigned to an Ignition tag that is tied to a PLC memory address, then the default value should be what is normal and default for the machine. However, it can be any value you want as long as it is within the range of the data type for the tag the recipe value is associated with and is within the security settings for the recipe value. See Recipe Security for more information.

Inside a recipe, a recipe value can use the default value or it can be overridden in the recipe as shown in the image below. Notice the Agitator Speed and Ingredient 2 of the Thick Blend recipe have been overridden. If it uses the default value, it will be updated when the default value is changed. Once a recipe value has been overridden in a recipe, it can then be reverted back to the default value.

© Inductive Automation

Default Values

Mixer	
Value Name	Default Value
Agitator Speed	35
Mix Time	20
Ingredient 1	15
Ingredient 2	50.2
Ingredient 3	8.2

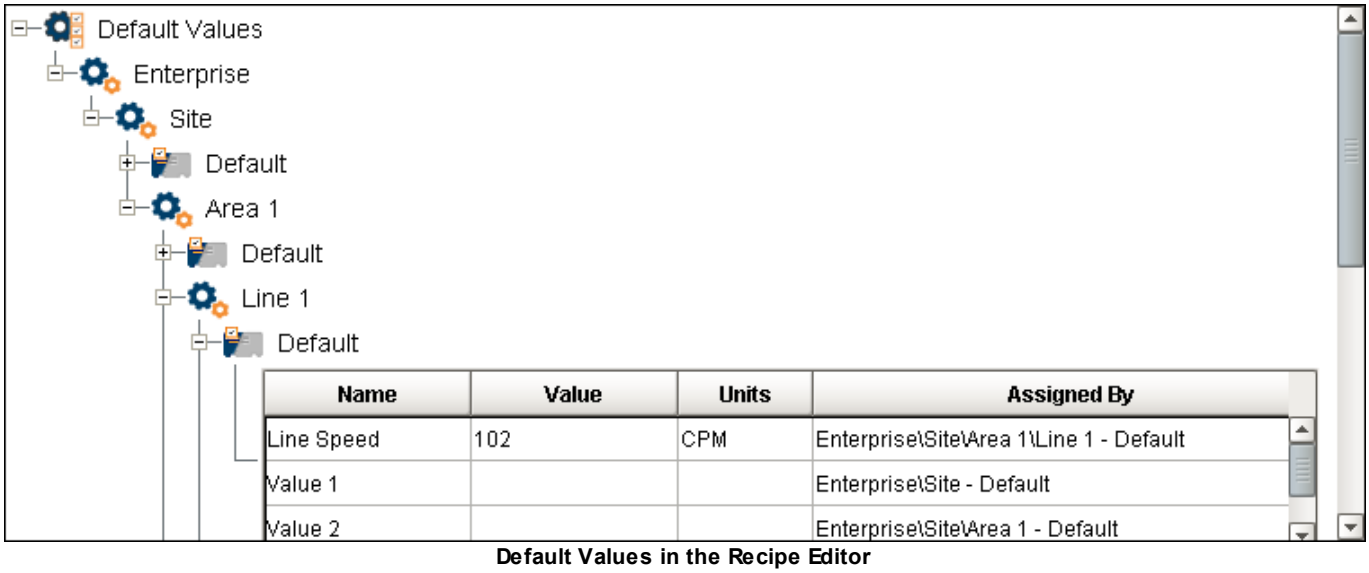
Recipes

Creamy Blend		
Value Name	Recipe Value	Assigned By
Agitator Speed	35	Mixer-Default
Mix Time	20	Mixer-Default
Ingredient 1	15	Mixer-Default
Ingredient 2	50.2	Mixer-Default
Ingredient 3	8.2	Mixer-Default

Thick Blend		
Value Name	Recipe Value	Assigned By
Agitator Speed	**37**	**Thick Blend**
Mix Time	20	Mixer-Default
Ingredient 1	15	Mixer-Default
Ingredient 2	**55.0**	**Thick Blend**
Ingredient 3	8.2	Mixer-Default

Recipe Default Values

All of this is done in the recipe editor or by using script functions. The image below shows the default values in the recipe editor where the default values can be edited. Also see Sub Recipes for more details of how default values are use with sub recipes. The section on Recipe Security provides more detail on changing the security for recipe values that are accessed in the default values.



Default Values in the Recipe Editor

4.1.4 Master Recipes

Master Recipes

Making a change to a recipe value that is used in numerous recipes is a daunting task and is prone to mistakes. To address this problem the Recipe / Changeover Module uses master recipes. The image below shows two recipes that are derived from, or descendents of, the Master Blend recipe. When the descendant recipe is added, all recipe values will be inherited from the master recipe. When a value is changed in a descendant recipe, it will override the value from the master recipe with the new value as shown in the image below for the Agitator Speed and Ingredient 2 recipe values.

The image below just shows one master recipe and two descendant recipes. In actual fact, there can be any number of levels of master recipes and any number of descendants of a master recipe. Any recipe that has descendants is considered a master recipe. Consider a master recipe called Master 1 that has a descendant that is called Master 1-A that has a descendant called Final 1-A-A. Then recipes Master 1 and Master 1-A are both master recipes and recipe Final 1-A-A is a final recipe. Only final recipes can be selected for a production line, cell, cell group or location. See Selecting Recipes for more information.

One aspect that is not shown in the image below is that the master recipe can inherit its values from the default values of the associated production item. So the production item has its defaults values, which is added to a recipe so the recipe inherits from the default values, then the descendant recipes inherit from the master recipe and so on. That is until a recipe value is overridden somewhere along the inheritance chain. See Default Values for more information.

Master Recipe

Master Blend		
Value Name	Recipe Value	Assigned By
Agitator Speed	35	Master Blend
Mix Time	20	Master Blend
Ingredient 1	15	Master Blend
Ingredient 2	50.2	Master Blend
Ingredient 3	8.2	Master Blend

Descendant Recipes

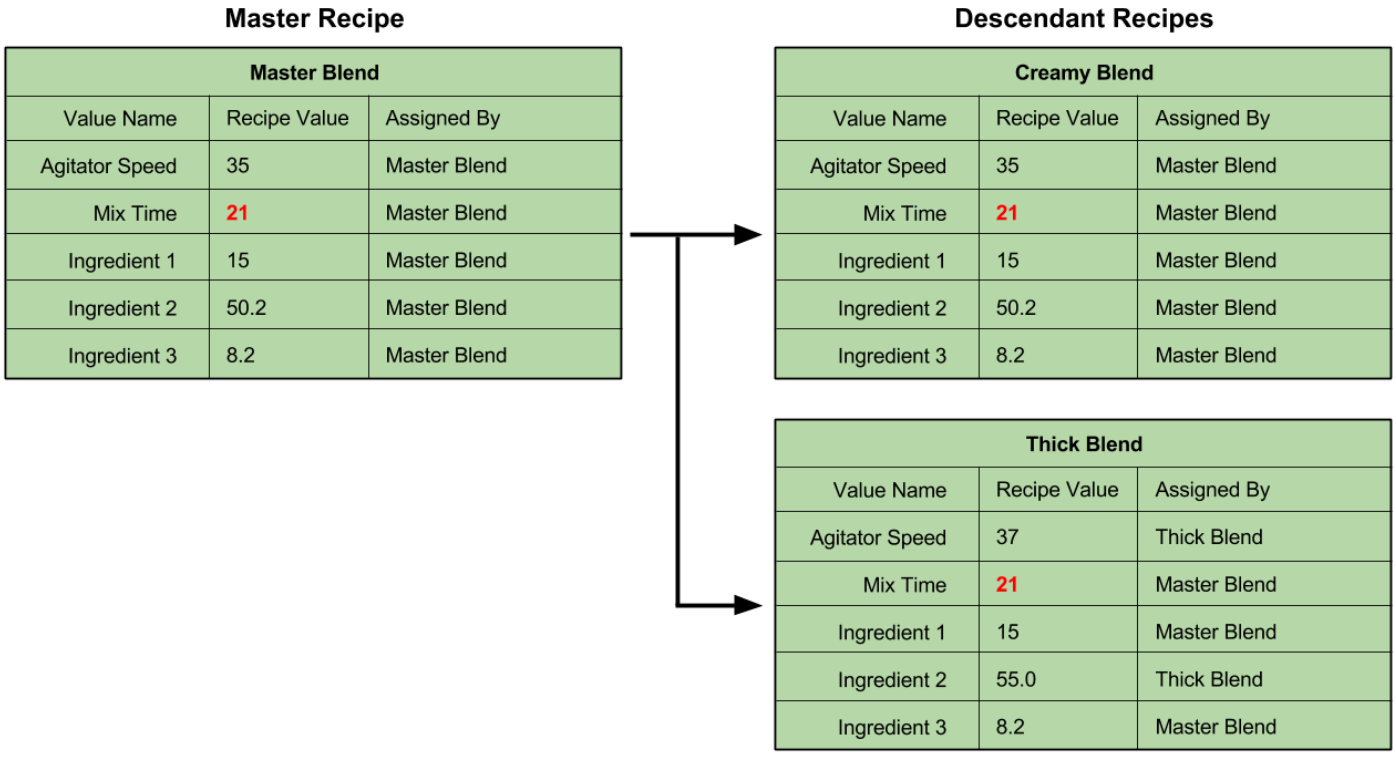
Creamy Blend		
Value Name	Recipe Value	Assigned By
Agitator Speed	35	Master Blend
Mix Time	20	Master Blend
Ingredient 1	15	Master Blend
Ingredient 2	50.2	Master Blend
Ingredient 3	8.2	Master Blend

Thick Blend		
Value Name	Recipe Value	Assigned By
Agitator Speed	37	Thick Blend
Mix Time	20	Master Blend
Ingredient 1	15	Master Blend
Ingredient 2	55.0	Thick Blend
Ingredient 3	8.2	Master Blend

Master Recipe

When a value is changed in the master recipe, it is propagated down to the descendant recipes. As shown in the image below, the Mix Time recipe value is changed to 21 and the Creamy Blend and Thick Blend recipes also reflect the new value.

© Inductive Automation


Master Recipe
Descendant Recipes
Master Blend
Value Name Recipe Value Assigned By
Agitator Speed 35 Master Blend
Mix Time 21 Master Blend
Ingredient 1 15 Master Blend
Ingredient 2 50.2 Master Blend
Ingredient 3 8.2 Master Blend
Creamy Blend
Value Name Recipe Value Assigned By
Agitator Speed 35 Master Blend
Mix Time 21 Master Blend
Ingredient 1 15 Master Blend
Ingredient 2 50.2 Master Blend
Ingredient 3 8.2 Master Blend
Thick Blend
Value Name Recipe Value Assigned By
Agitator Speed 37 Thick Blend
Mix Time 21 Master Blend
Ingredient 1 15 Master Blend
Ingredient 2 55.0 Thick Blend
Ingredient 3 8.2 Master Blend

Master Recipe Value Change

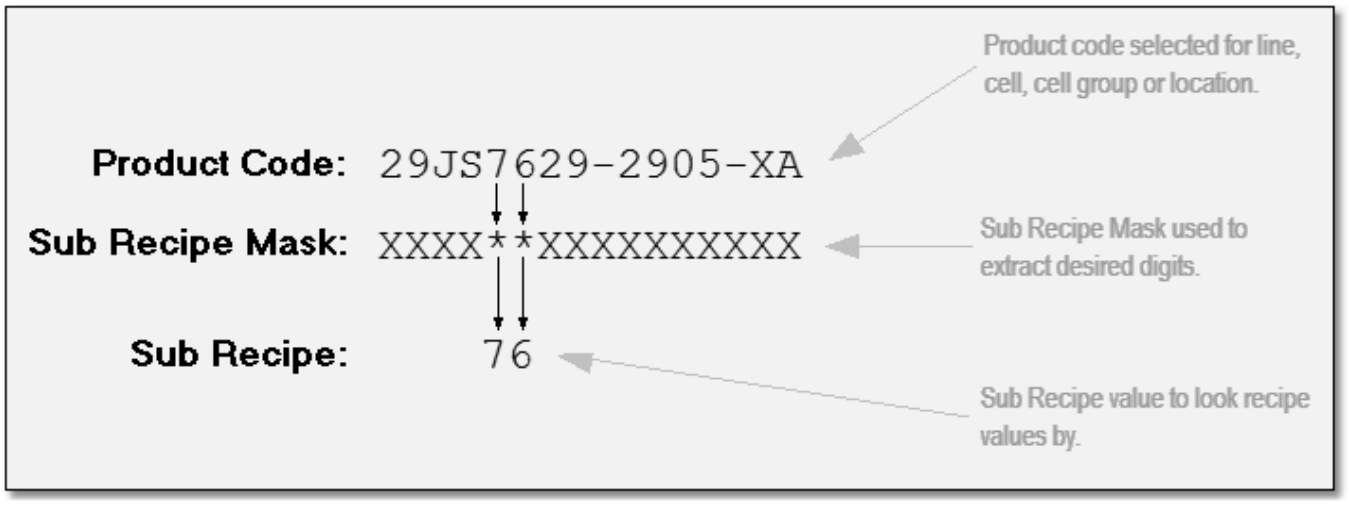
4.1.5 Sub Recipes

Sub Recipes

Sub Recipes are convenient when a machine's recipe can be determined from digits within a product code. When the product code is used as the recipe, a portion of the product code can be extracted and used to determine the machine's recipe. For example, you may have a tape machine that recipe values only change based on the case size. If there are only two different case sizes and there is a digit in the product code that specifies the case size, then sub recipes can be used for the tape machine. All other machines can use the normal recipe functionality.

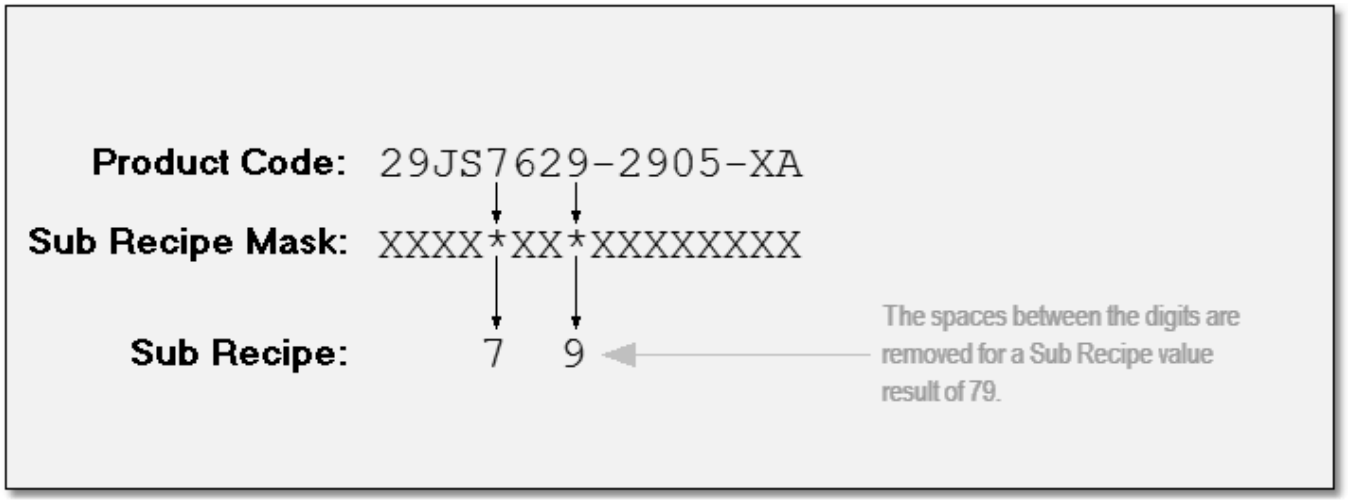
Sub recipes are derived from the product code and the sub recipe mask. The sub recipe mask specifies the digits to extract from the product code to determine the sub recipe. Once the sub recipe value is determined like the 76 in the image below, the recipe values are looked up in the sub recipes for the production line, cell, cell group or location and are written to tags. See Sub Recipe Mask for more information on how to configure production items to use sub recipes.

© Inductive Automation


Product code selected for line, cell, cell group or location.
Product Code: 29JS7629-2905-XA
Sub Recipe Mask: XXXX**XXXXXXXXXX
Sub Recipe Mask used to extract desired digits.
Sub Recipe: 76
Sub Recipe value to look recipe values by.

Sub Recipe Mask

There are two different mask characters that should be used in the Sub Recipe Mask. The first is just a placeholder and any characters that exist in the corresponding digit position of the product code will be ignored. The other mask character is the asterisk and any characters that exist in the corresponding digit position of the product code will be used in the sub recipe value. The asterisk characters in the Sub Recipe Mask do not need to be in consecutive digit positions as shown in the image below.

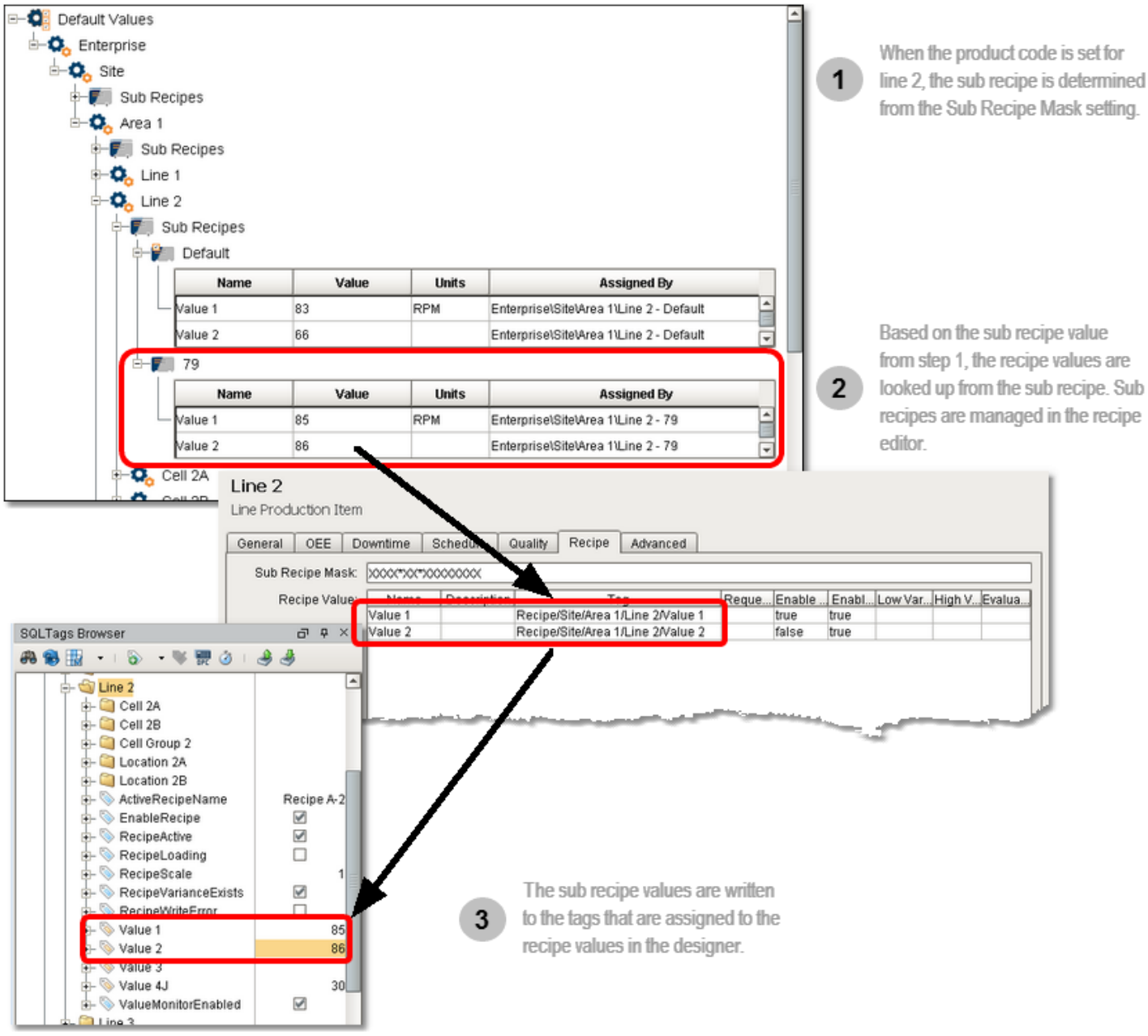

Product Code: 29JS7629-2905-XA
Sub Recipe Mask: XXXX*XX*XXXXXXXX
Sub Recipe: 7 9
The spaces between the digits are removed for a Sub Recipe value result of 79.

Sub Recipe Mask with Non-Consecutive Asterisk Mask Characters

The image below steps through the flow of selecting sub recipes and setting the associated tag values and is based on the determining of the sub recipe as described above. The product code can be selected using various methods. It can be selected using the Recipe Selector List component, but is can also be selected by starting a production run for the OEE Module or by using one of the script functions. In fact, it can be selected using a combination of methods.

The Recipe Editor component is used to edit both normal recipes and sub recipes. See sub recipes in the Editing Recipes more details. In general, the recipe editor is used to manage sub recipes for a line, cell, cell group or location. New sub recipes can be created and the recipe values for each can be edited.

© Inductive Automation


Default Values
Enterprise
Site
Sub Recipes
Area 1
Sub Recipes
Line 1
Line 2
Sub Recipes
Default
Name Value Units Assigned By
Value 1 83 RPM Enterprise\Site\Area 1\Line 2 - Default
Value 2 66 Enterprise\Site\Area 1\Line 2 - Default
79
Name Value Units Assigned By
Value 1 85 RPM Enterprise\Site\Area 1\Line 2 - 79
Value 2 86 Enterprise\Site\Area 1\Line 2 - 79
Cell 2A
1
When the product code is set for line 2, the sub recipe is determined from the Sub Recipe Mask setting.
2
Based on the sub recipe value from step 1, the recipe values are looked up from the sub recipe. Sub recipes are managed in the recipe editor.
Line 2
Line Production Item
General OEE Downtime Schedu Quality Recipe Advanced
Sub Recipe Mask:
Recipe Value:
Value 1 Recipe/Site/Area 1/Line 2/Value 1 true true
Value 2 Recipe/Site/Area 1/Line 2/Value 2 false true
SQLTags Browser
Line 2
Cell 2A
Cell 2B
Cell Group 2
Location 2A
Location 2B
ActiveRecipeName Recipe A-2
EnableRecipe
RecipeActive
RecipeLoading
RecipeScale 1
RecipeVarianceExists
Value 1 85
Value 2 86
Value 4J 30
ValueMonitorEnabled
3
The sub recipe values are written to the tags that are assigned to the recipe values in the designer.

If a sub recipe is not found, the default will be used.

4.1.6 Editing Recipes

There are multiple methods that values of a recipe can be changed. Depending on the functionality that you are looking for, recipe values can be changed using the recipe editor, by importing or from script.

Recipe Editor

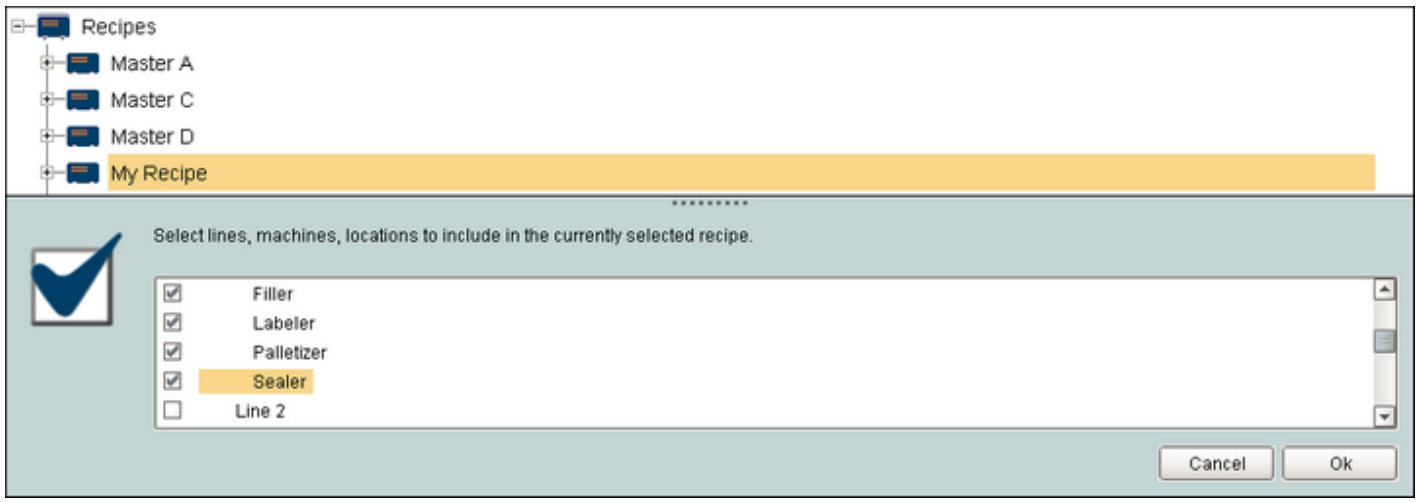
The recipe editor component provides a visual and interactive method to allow end users to manage recipes. It handle all of the details and is as easy as adding the component to any Ignition window. It also provides the ability to manage sub-product codes, recipe value security, master recipes and add MES production items to recipes.

To add a new recipe, right click on the root Recipes item in the recipe editor and select Add Recipe menu item. The new recipe will be added and will be ready to enter the name of the new recipe. Commonly, the

name of the recipe will be the same as a product code, but it does not have to be. It can represent a mode of the machine such as Cleaning Mode.

Add Recipe

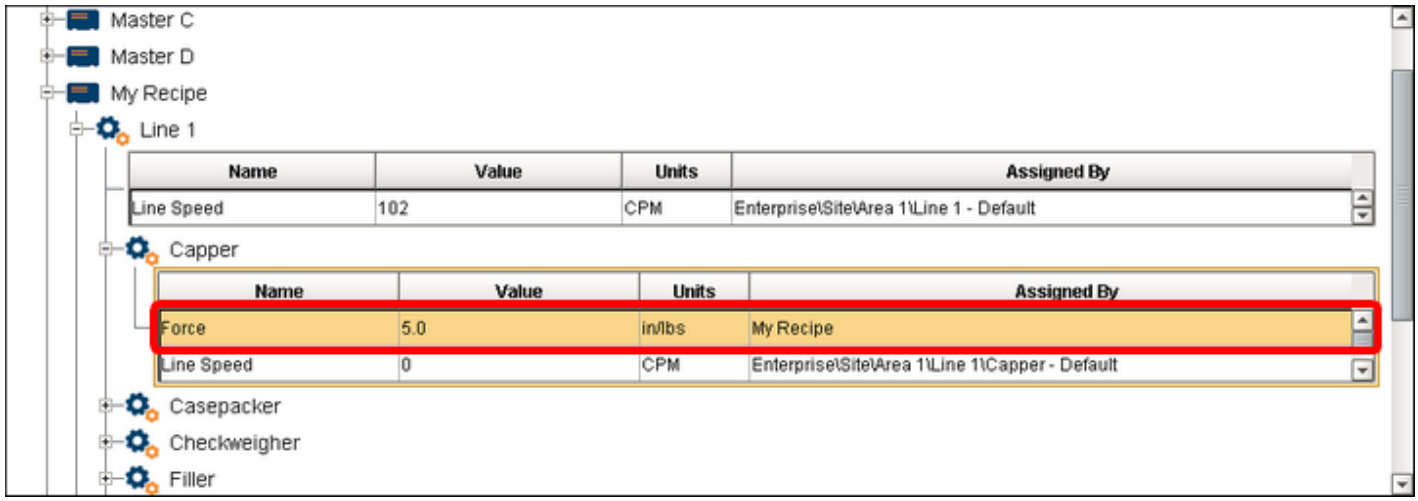
Type in the name of the new recipe which for this example it is My Recipe. Next, right click on the new My Recipe and click on the Select Production Items menu item. Please note, that you must first add production items in the designer before they appear as options to be added to a menu. Because not all machinery is used in every recipe, this step is used so that only the machinery that is appropriate for a recipe appears in the recipe editor and recipe selector components. For this example, Line 1 and all of the cells (machines) beneath it are added to the recipe.



Select Production Items to Add to Recipe

After clicking the Ok button, expand the production item to view and edit the recipe values. Notice the Assigned By column. When the new recipe was first added, all of the recipe values show a assigned by of Enterprise\Site\Area 1\Line 1\Capper - Default for the capper. This is because the initial values of a new recipe are inherited from the default values for the capper production item. See Default Values for more information. When a recipe value is changed, the assigned by changes to My Recipe. This is because the value is no longer that from the default values and is now from the recipe. In simpler terms, it tells you where the value has been changed in the inheritance tree.

© Inductive Automation



Edit Recipe Values

My Recipe will now appear in the Recipe Selection List component and can be selected for any of the production items that were added to the recipe. But, My Recipe can also be made into a master recipe simply be add descendant recipes to it. See Master Recipes for more information. This is done by right clicking on the Descendants item beneath the My Recipe recipe in the recipe editor, and clicking on the Add Recipe menu item. Type in the name of the new recipe which for this example it is My Recipe 1. There is no limit to the number of descendants recipes you can add to a master recipe. There is also no limit to the number of levels deep of master recipes. After My Recipe 1 is added to My Recipe, My Recipe will no longer show as a option in the recipe selection list component but My Recipe 1 will. In general, if a recipe has descendants, it becomes a master recipe and will no longer show in the recipe selection list component. Only final recipes with no descendant will show in the recipe selection list component.

However, master recipes can be selected for production items by using script functions.

Import / Export

Recipes values can be imported and exported into the recipe management system. Note that only the actual value of the recipe value item can be imported and not the recipe value definition. Recipe value definitions can be imported in the designer. See Recipe Value Import / Export for more information. This is because the recipe value definitions are tightly tied to production items (equipment) and tags, both of which cannot be created in the client.

To export the recipe values in the client, right click on a line, cell, cell group or location underneath the line and select the export menu item. A file chooser dialog will appear to select or enter a file name to export to. The file format is a comma separated values (CSV) and contains the following columns:

- Recipe_Name
- Value_Name
- Item_Path
- Description
- Units
- Data_Type
- Format
- Recipe_Value
- Assigned_By

© Inductive Automation

To import recipe values in the client, right click on a line, cell, cell group or location underneath the line and select the import menu item. A file chooser dialog will appear to select or enter a file name to import to. The file format is a comma separated values (CSV) and must contain the following columns:

- Recipe_Name
- Value_Name
- Item_Path
- Recipe_Value

All other columns will be ignored during the import. Also, all values must be surrounded with quotes including the recipe value. During importing, the recipe value will be converted to the appropriate data type that the recipe value is defined as, which is based on the tag it is associated.

Recipe values can be imported for multiple recipes and production items combinations in one import operation as defined with the Recipe_Name and Item_Path columns of the CSV file. This supports bulk import operations instead of only being limited to one recipe at a time.

Recipe values can also be imported and exported using script either at the client or in the gateway. See exportRecipe and importRecipe script functions for more information. The following is an example statement that will import recipe values on the gateway. The project name is required because recipe values are managed by project. The csvData parameter is a string of csv data that can be read in from a file, web service, etc. And last, the note is what will show in the recipe change log.

```
system.recipe.importRecipe("RecipeProject", csvData, note)
```

This functionality supports reading recipe values from ERP or other systems that are currently being used to manage recipes. Once the recipe values are in Ignition they can be selected, monitored for variances, analysis, etc.

Script

In addition to importing and exporting recipe values there are scripts to add recipes, rename recipes, delete recipes and much more. See Client / Gateway Scripts for documentation of all script functions. Because the built-in functionality will not fit the requirements in every situation, the scripting functions provide the built-in functionality to be extended to accommodate the requirements.

4.1.7 Recipe Change Log

Keeping an audit log of when recipes are changed, by who and why can be important. Especially in some industries where regulatory compliance are in force. The Recipe / Changeover Module records all changes to recipes wether the changes were made from the recipe editor, importing or script. The only changes not automatically detected are changes made directly to the database and proper database security should be implemented if this is a risk.

Below are the methods that the change log history can be examined.

Recipe Change Log Viewer

There is a component that will easily show recipe change log history on screens. It has properties to narrow in on what production item and recipes to show the change log for. The columns that are shown are configurable through the table customizer. See Recipe Change Log Viewer component for more information. The image below depicts the change log viewer.

© Inductive Automation

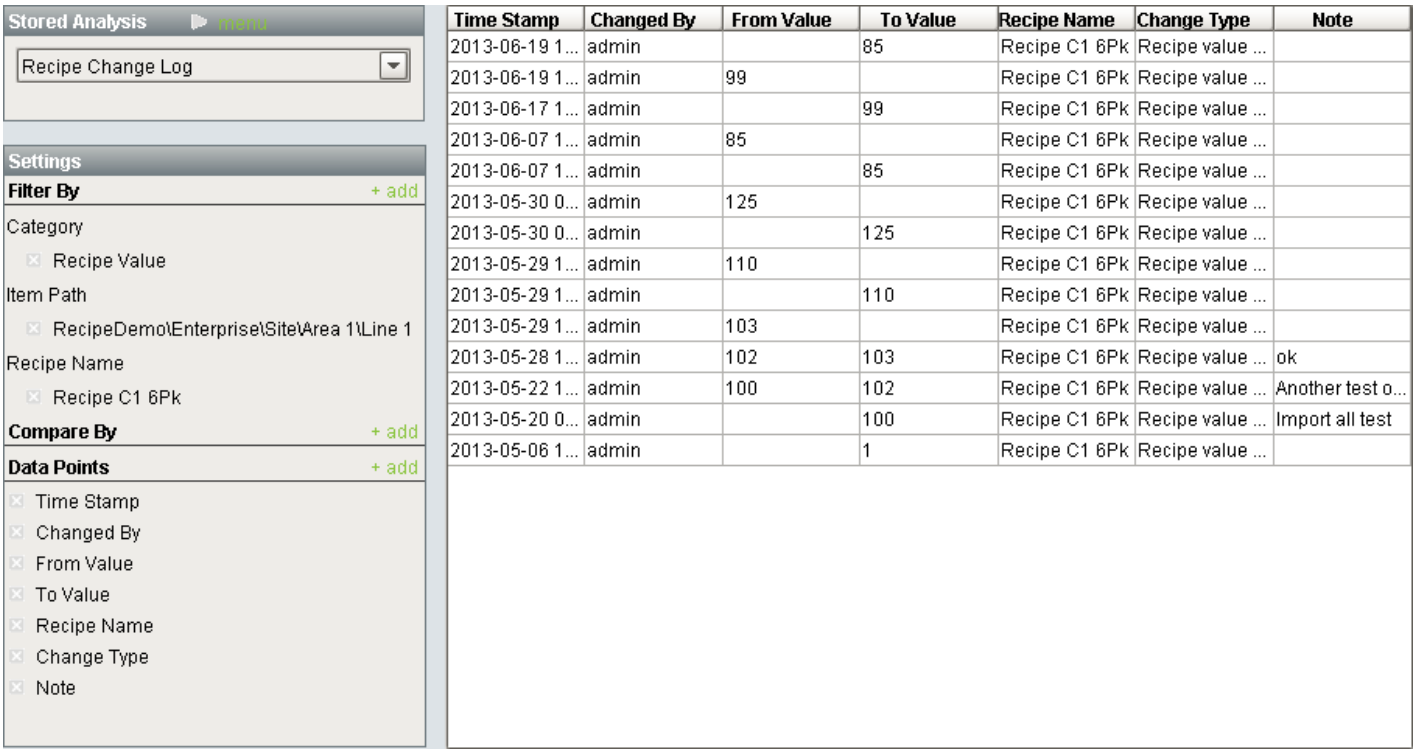
ValueName	Change...	TimeStamp	ChangeType	FromValue	ToValue
Line Speed	admin	Jun 19, 2013 3:18 PM	Recipe value changed		85
Line Speed	admin	Jun 19, 2013 3:17 PM	Recipe value reverted	99	
Line Speed	admin	Jun 17, 2013 11:21 AM	Recipe value changed		99

Recipe Change Log Viewer

Recipe Change Log Analysis Provider

The core production module common to all MES modules has a analysis engine. Each MES module provides a analysis provider to work with the data collected for the specific module. In the case of the Recipe / Changeover Module, there are 4 different analysis providers of which one of them is the Recipe Change Log provider. It is used to query change log history information based on your selections. See Recipe Analysis Provider for more information.

The image below shows the interactive Recipe Change Log Analysis Provider using the core analysis components, but it can also be used to provide data to the Ignition Reporting Module.


Stored Analysis
menu
Recipe Change Log
Settings
Filter By
+ add
Category
Recipe Value
Item Path
RecipeDemo\Enterprise\Site\Area 1\Line 1
Recipe Name
Recipe C1 6Pk
Compare By
+ add
Data Points
+ add
Time Stamp
Changed By
From Value
To Value
Recipe Name
Change Type
Note
Time Stamp | Changed By | From Value | To Value | Recipe Name | Change Type | Note
2013-06-19 1... | admin | | 85 | Recipe C1 6Pk | Recipe value ... |
2013-06-19 1... | admin | 99 | | Recipe C1 6Pk | Recipe value ... |
2013-06-17 1... | admin | | 99 | Recipe C1 6Pk | Recipe value ... |
2013-06-07 1... | admin | 85 | | Recipe C1 6Pk | Recipe value ... |
2013-06-07 1... | admin | | 85 | Recipe C1 6Pk | Recipe value ... |
2013-05-30 0... | admin | 125 | | Recipe C1 6Pk | Recipe value ... |
2013-05-30 0... | admin | | 125 | Recipe C1 6Pk | Recipe value ... |
2013-05-29 1... | admin | 110 | | Recipe C1 6Pk | Recipe value ... |
2013-05-29 1... | admin | | 110 | Recipe C1 6Pk | Recipe value ... |
2013-05-29 1... | admin | 103 | | Recipe C1 6Pk | Recipe value ... |
2013-05-28 1... | admin | 102 | 103 | Recipe C1 6Pk | Recipe value ... | ok
2013-05-22 1... | admin | 100 | 102 | Recipe C1 6Pk | Recipe value ... | Another test o...
2013-05-20 0... | admin | | 100 | Recipe C1 6Pk | Recipe value ... | Import all test
2013-05-06 1... | admin | | 1 | Recipe C1 6Pk | Recipe value ... |

Recipe Change Log Analysis

Script

Script functions can also be used to read the recipe change log history. This is useful if the recipe change log history is needed for reasons other than displaying or reporting. The recipe change log history is returned as a dataset from the script and the rows can be iterated through.

© Inductive Automation

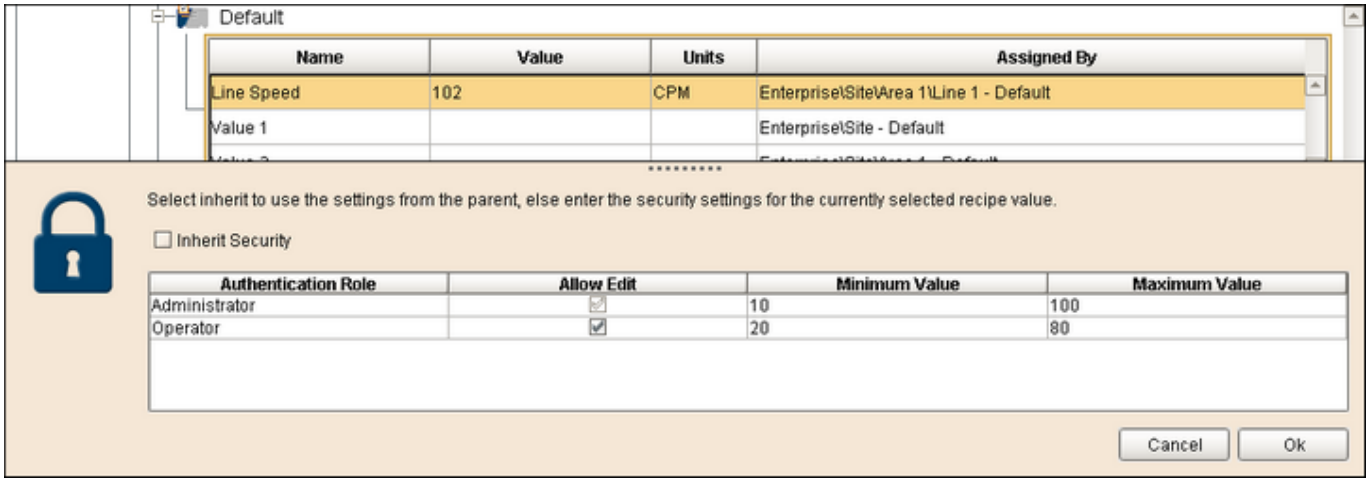
See getChangelogHistory script function for more information.

4.1.8 Recipe Security

Recipe Value Security

The recipe value security uses Ignition's authentication roles to limit who can change what recipe values by how much. Each recipe value can be set to specific security settings or it can inherit from its parent. Like other recipes value settings, the security settings can propagate down multiple levels of inheritance.

Referring to the image below, the Inherit Security check box determines if the recipe value should use its parent's security settings or break the inheritance. By unselecting the Inherit Security check box, the settings for each authentication role can be made. Initially when doing so, the inherited security settings will remain that of the parent until they are edited.



Recipe Value Security Settings

The recipe value security is verified when changing values using the recipe editor component, importing recipes or changing values using script.

When changing a recipe value using the recipe editor component, importing recipe values or from client script, the authentication role applied comes from the roles the currently logged in user belongs to. If the user belongs to multiple roles then the role with the least security will be applied. For example, if a user belongs to both the **Operator** and **Maintenance** authentication roles, then the least secure one will be applied. If the **Operator** role can change the Product Pressure recipe value from 10 to 15 and the **Maintenance** role can change it from 5 to 20, then the **Maintenance** role will apply.

When changing a recipe value from gateway script, the **Administrator** authentication role is always applied.

The only place the recipe value security can be changed is by using the recipe editor component. Also, it can only be changed in the default values area and not in the actual recipes. Wether or not the logged in user can change the security settings can be controlled with the Enable Security Editing property of the recipe editor component. This property can be bound to an expression to determine if the currently logged in user belongs to authentication roles that are allow to edit security. Another approach is to create a window that allows the recipe value security editing and restrict opening the windows based on authentication roles the currently logged in user belongs to.

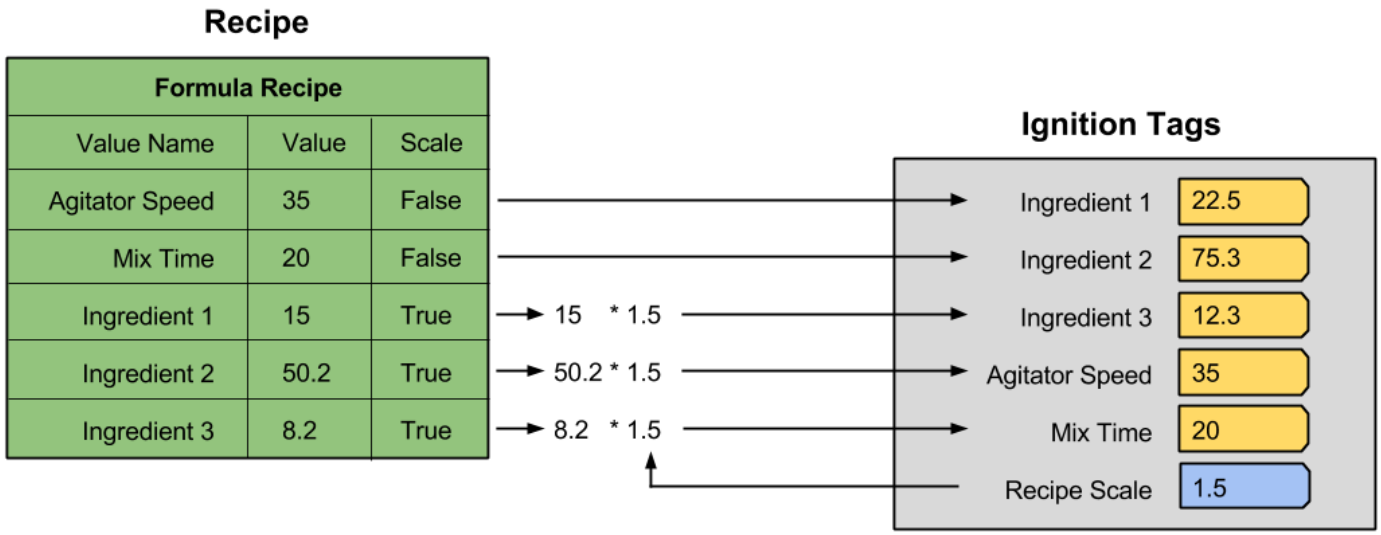
© Inductive Automation

4.1.9 Recipe Scaling

Recipe Scaling

When using recipes for batch or other processes that can change based on the amount that is produced, recipe scaling will adjust recipe values based on a recipe scale value. In the recipe value configuration, there is an Enable Scaling setting that can be selected. If the Enable Scaling setting is selected for a recipe value, then whenever a recipe is selected for a production line, cell, cell group or location, the value from the recipe will be scaled by the value in the RecipeScale tag as shown in the image below.

Enabling recipe scaling is done for each individual recipe value. This supports scaling some of the recipe values while not scaling others as might be the case in the example shown below. By default, each production item's Enable Scaling setting is false and must be selected before the RecipeScale value will be applied.



Recipe Scaling

The RecipeScale is a production OPC item that exists for each production line, cell, cell group or location. By default, the RecipeScale is 1.0 and recipe values will not change when recipes are selected. When selecting a recipe for a line, all of the cells, cell groups and locations beneath the line will also be set to the same recipe provided they are enabled. Also, each cell, cell group and location RecipeScale value will be set to match that of the line. This enables simple recipe selection for a line without the tedious task of selecting each machine underneath the line.

4.1.10 Variance Monitoring

Variance Monitoring

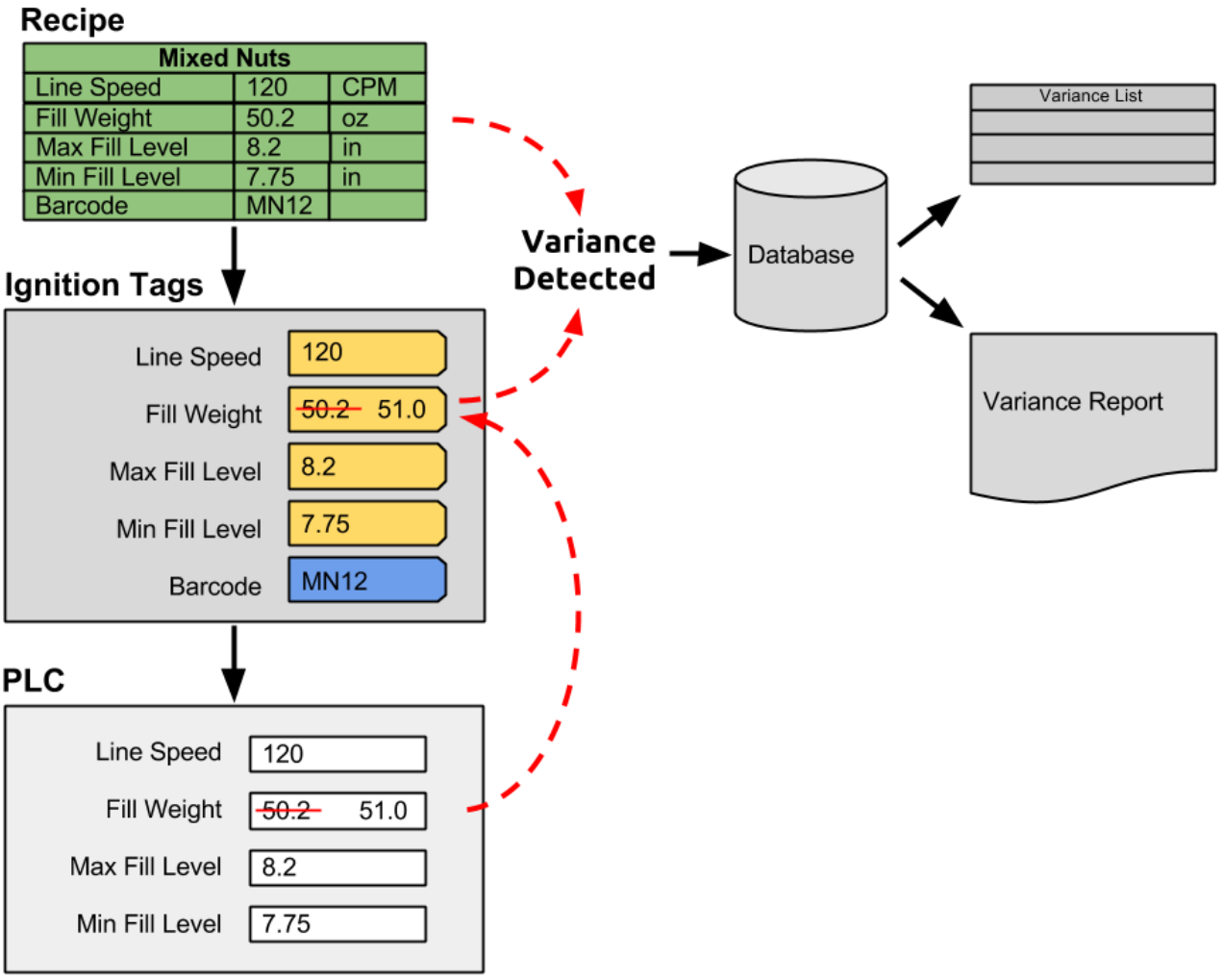
In most manufacturing systems it is important to know if the live production values match the recipe values. There are two cases where this is important. The first is when the recipe values are first written to verify that they match. The second is during production in the event the live production values changed from an outside source.

Recipe values are written once when the recipe is first selected. It is very important to confirm that the values were successfully set. In the case of the Recipe / Changeover Module, when a recipe is selected, the values are written to the Ignition tags. This should happen successfully, but there can be expressions, scripts, etc. that prevent the value from being written correctly. This is more of an issue when the Ignition tag is configured as a OPC item connecting it to the PLC or other device. If a communication error occurs

© Inductive Automation

when the new recipe value was being written to the PLC or device, then it is very useful to know this before machinery is started.

It is very common to have operator interface terminals (OIT) or a standalone human machine interface (HMI) local to a machine that settings can be changed locally. Settings can also be changed from other sources besides the local OIT, and it is important to detect and log when any setting varies from the recipe value.



Recipe Variance Detection

Recipe Value Variance Options

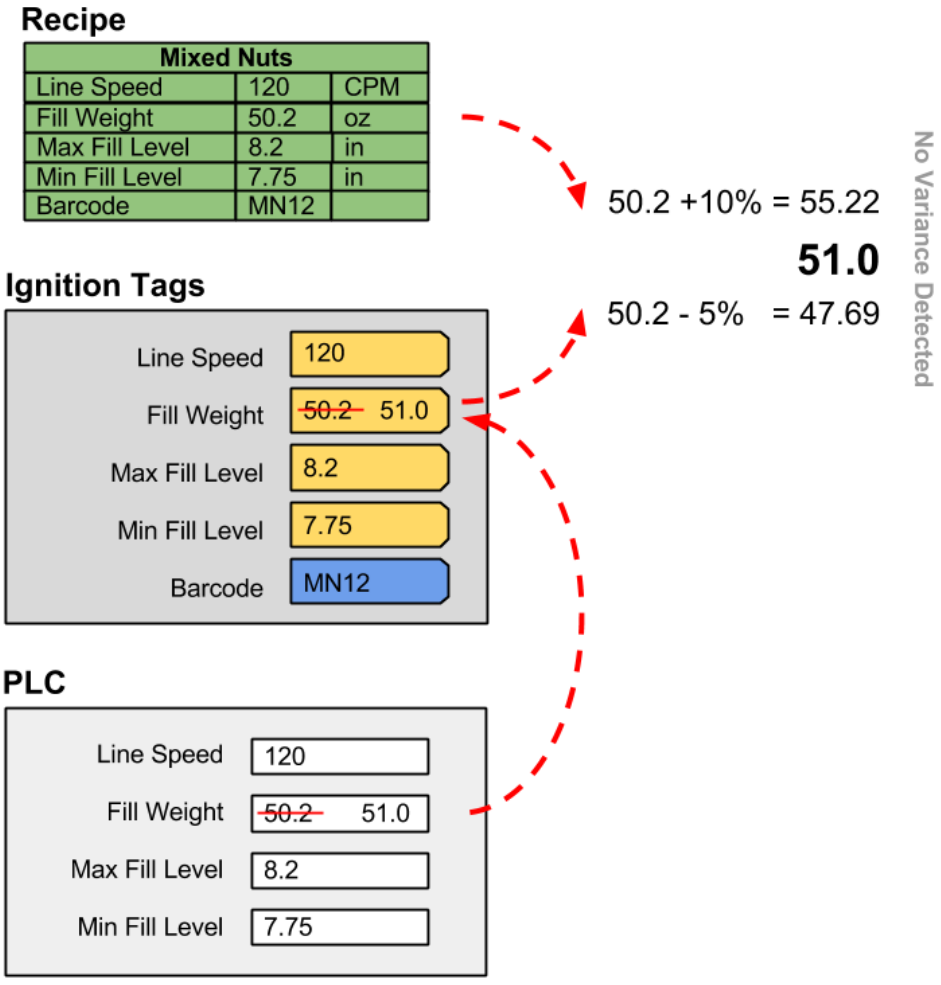
There are cases where it is normal for a live production value to vary after the initial recipe value has been written to the Ignition tag. In other cases, it might be okay for the live production value to change within a range. Recipe values in the Recipe / Changeover Module can be configured to not monitor variances or to have a variance window that the live production value must fall outside of before the variance is logged.

By default the variance monitoring is enabled for each recipe value but it can be disabled by recipe value in the designer. This allows for a mix of recipe values that variances will be monitored and other that will not to prevent irrelevant variances from being logged.

The configuration for the variance window is also done by recipe value in the designer. Both the upper and

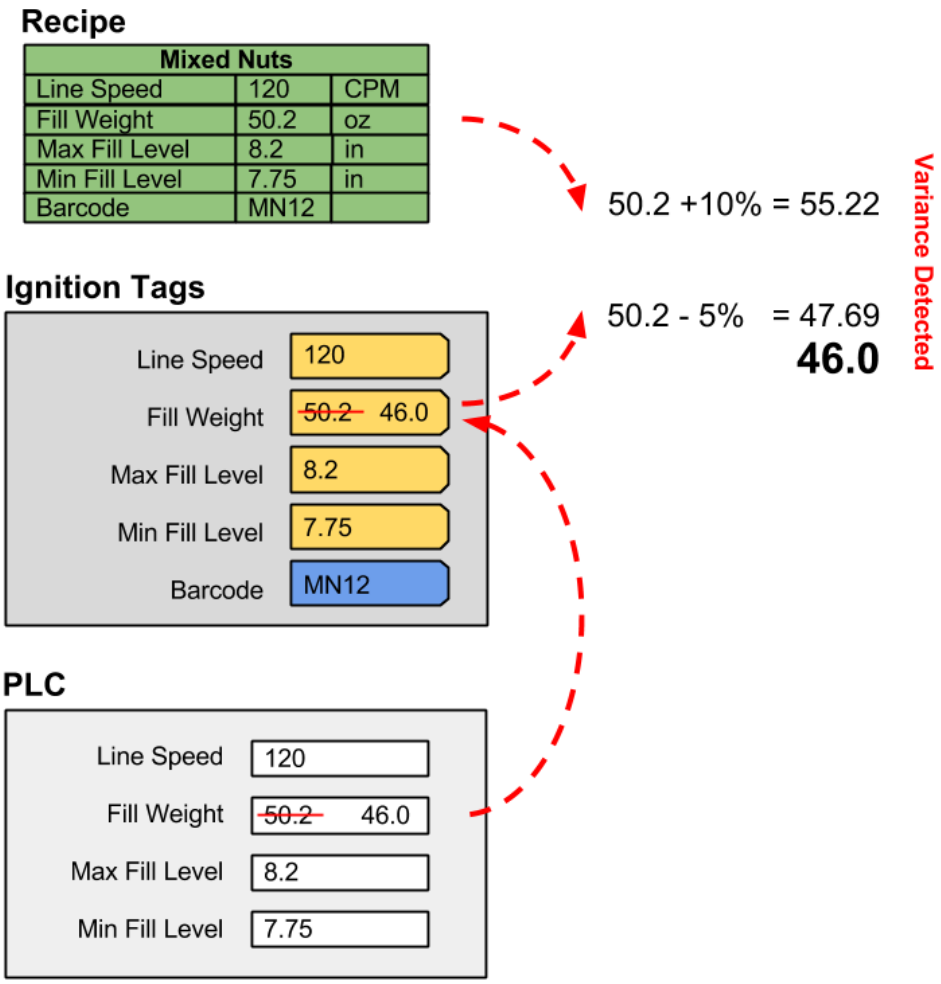
© Inductive Automation

lower variance thresholds can be defined by percentage of the recipe value or a fixed offset around the recipe value or just fixed values. The image below is showing the Fill Weight recipe value with an upper variance threshold of +10%. The upper threshold value is calculated by starting with the recipe value of 50.2 and adding 10%. The lower threshold is calculated in the same manner. When the Ignition tag value changes, a check is done to see if the current value is between the upper and lower threshold values and in this case as shown in the image below, we see that the current value of 51.0 is between 55.22 and 47.69. As a result, no variance will be logged.



Recipe Value Inside Range

Now lets take a look at a case where the current value is 46.0 as shown in the image below. The value 46.0 is less than the lower threshold and a variance will be logged.



Recipe Value Outside Range

The example above shows the upper and lower threshold values being calculated as a percentage of the recipe value, but they can also be a fixed offset around the recipe value. To configure a recipe value for a fixed offset around the recipe value a upper variance threshold setting of +<offset> is used. An example is a variance threshold offset of +7.5 were the upper threshold is calculated by adding 7.5 to the recipe value. Using the recipe value from above of 50.2 and adding 7.5 to it will give us a upper threshold of 57.7. The lower variance threshold works the same way.

Instead of the thresholds being calculated as a percentage or fixed offset around the recipe value, fixed values can also be used. For example, a recipe value can be configured with a upper variance threshold of 52.0. In this case, the upper threshold will always be 52.0 irregardless of the recipe value.

Lastly, the upper and / or lower threshold can be calculated using Python script. This is configured in the designer and a recipe value variance range can refer to other tag values, values from databases and much more when calculating the upper or lower threshold values.

Example Evaluate Variance Script

```
upperValue = system.tag.read("[Default]SomeOtherTag")
recipeValue = event.getRecipeTag().getCurrentValue()
if recipeValue > upperValue.value:
    event.setLogVariance(True)
else:
```

```
event.setLogVariance(False)
```

The script is passed an Evaluate Variance Script object that allows accessing the current tag information and also allows setting the log variance flag. In the script above, a tag called SomeOtherTag is read and compared to the current value of the tag associated with the recipe value where this script was defined. If the current value is greater than the value of the SomeOtherTag, then the setLogVariance method is called with True meaning the recipe value is in a variance state. Otherwise, false is returned.

See Adding a Recipe Value section for details about configuring recipe values.

Variance Status

As mentioned above, variances are logged to the database and can be viewed in the Recipe Variance Viewer component, analysis and reports. But, having a tag that indicates if any recipe values are in variance for a machine is useful. The Ignition MES Modules exposes current status and allows some control through the Production OPC Server. One of the status values provided by the Recipe / Changeover Module is the RecipeVarianceExists value. Each production line, cell, cell group and location has an associated RecipeVarianceExists value. If the value is false, then all live production values are within variance for the production item. If the value is true, then at least one recipe value of the production item is outside of its variance range. See Production OPC Server and Production OPC Values sections for available values the Recipe / Changeover Module provides.

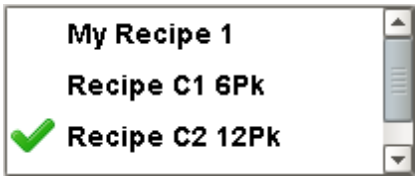
4.1.11 Selecting Recipes

There are different methods of selecting a recipe for production items. Production items that support recipe selection are lines, cells, cell groups and locations. When a production line recipe is selected, it will also select the same recipe for all of the cells, cell groups or locations that are children of the line. This can be disabled by setting the EnableRecipe tag for the child production item to false. This is a feature that makes day-to-day selection of line recipes easier and mistake free.

Recipes can also be canceled that simply turns off variance tracking. This will keep recipe value variance reporting clean with only data from actual production runs.

Components

After the Recipe / Changeover Module is installed, a Recipe tab will be added to component pallet in the Ignition designer. There are two components that allow selection of recipes for a production item. Below is what the Recipe Selection List component looks like.



Recipe Selection List Component

Only final recipes for the production item that is specified by the Item Path property will be displayed. See Master Recipes for more information about final recipes versus master recipes. The list can be limited to only show a subset of recipes by using the Recipe Name Filter property.

End users can select and cancel recipes using this component by right clicking on a recipe and selecting the desired menu item. If this functionality is not desired, then the Read Only property can be set to True to prevent the select popup menu from showing. See Recipe Selection List for more information about the

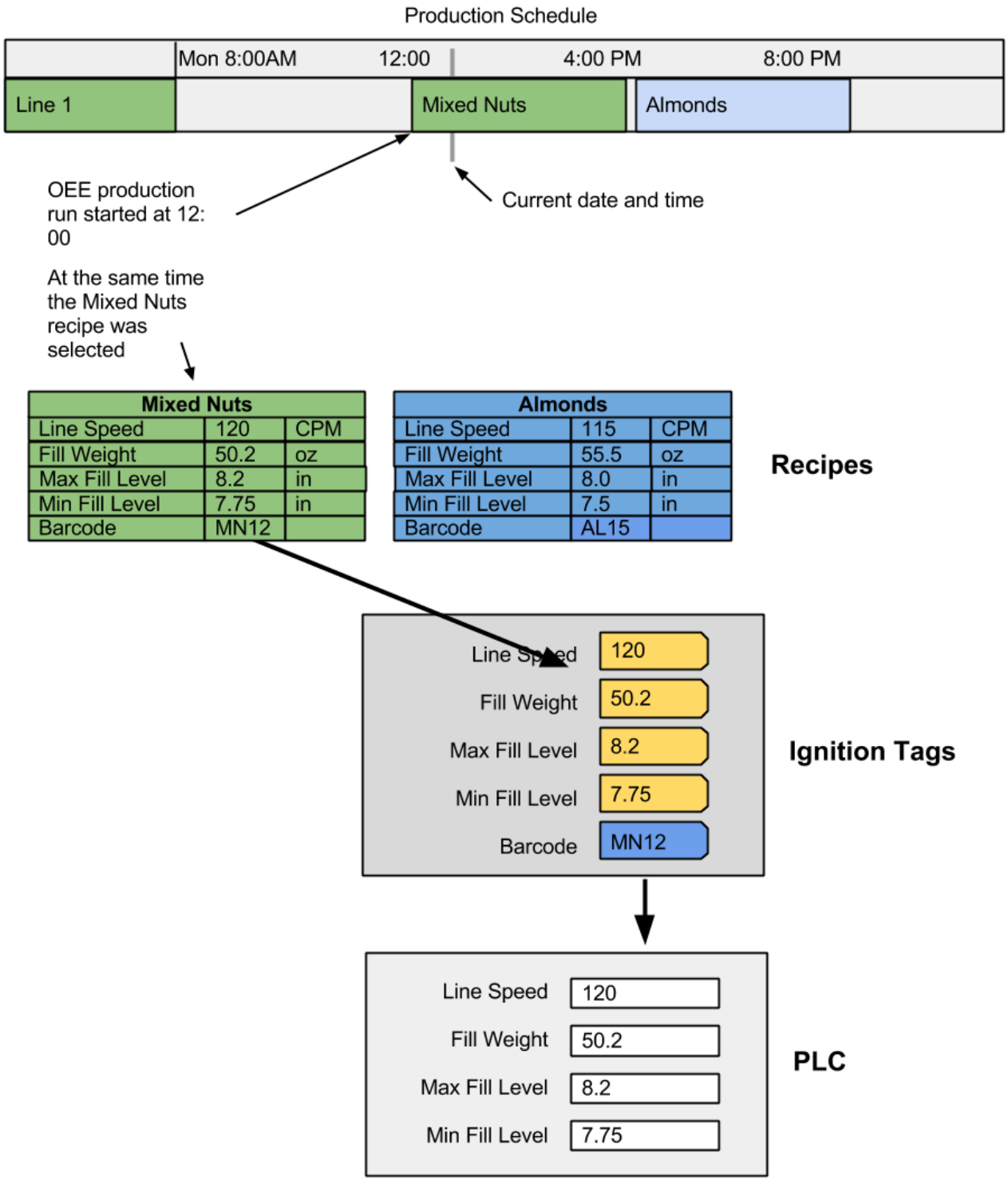
component.

Starting an OEE Production Run

When a production run is started for the OEE Downtime Module, a product code is selected (or a work order that is associated with a product code). If a recipe that is named exactly the same as the product code exists, and the production line that is being started for the OEE Downtime Module has been previously added to that recipe, then the recipe will be selected at the same time.

In the image below the Mixed Nuts production run started at 12:00 and at the same time the Mixed Nuts recipe was selected. This greatly simplifies setup of new production runs because only one selection of the product has to be made. This also eliminates mistakes of selecting different products for the production run and the recipe. OEE production runs can be started many different ways and using a production schedule as shown in the image is only one of them. For more information, see the MES OEE Downtime Module documentation.

© Inductive Automation



Recipe Selection when OEE Run Started

Starting a Trace

Production locations have the ability to trace product that is being process at them. When a production location product code is set using the setLocationProductCode script function, the recipe is also selected for the production location. When this happens the recipe values are written to their associated tags. The location trace can be turn off using the cancelLocationProductCode function.

Currently, scripting is the only method to select a production code for a location, but this will be enhanced when the traceability module is released.

Example

```
system.production.utils.setLocationProductCode("RecipeDemo\Enterprise\Site\Area 1\Filler Location", "
```

Scripts

Scripts can also be used to directly select and cancel recipes for production items. These scripts supports selecting and cancelling recipes from external triggers such as when a button is clicked or when a product code changes.

See setItemRecipe script function to select a recipe for a production item and cancelItemRecipe script function to cancel the current recipe for a production item for more details.

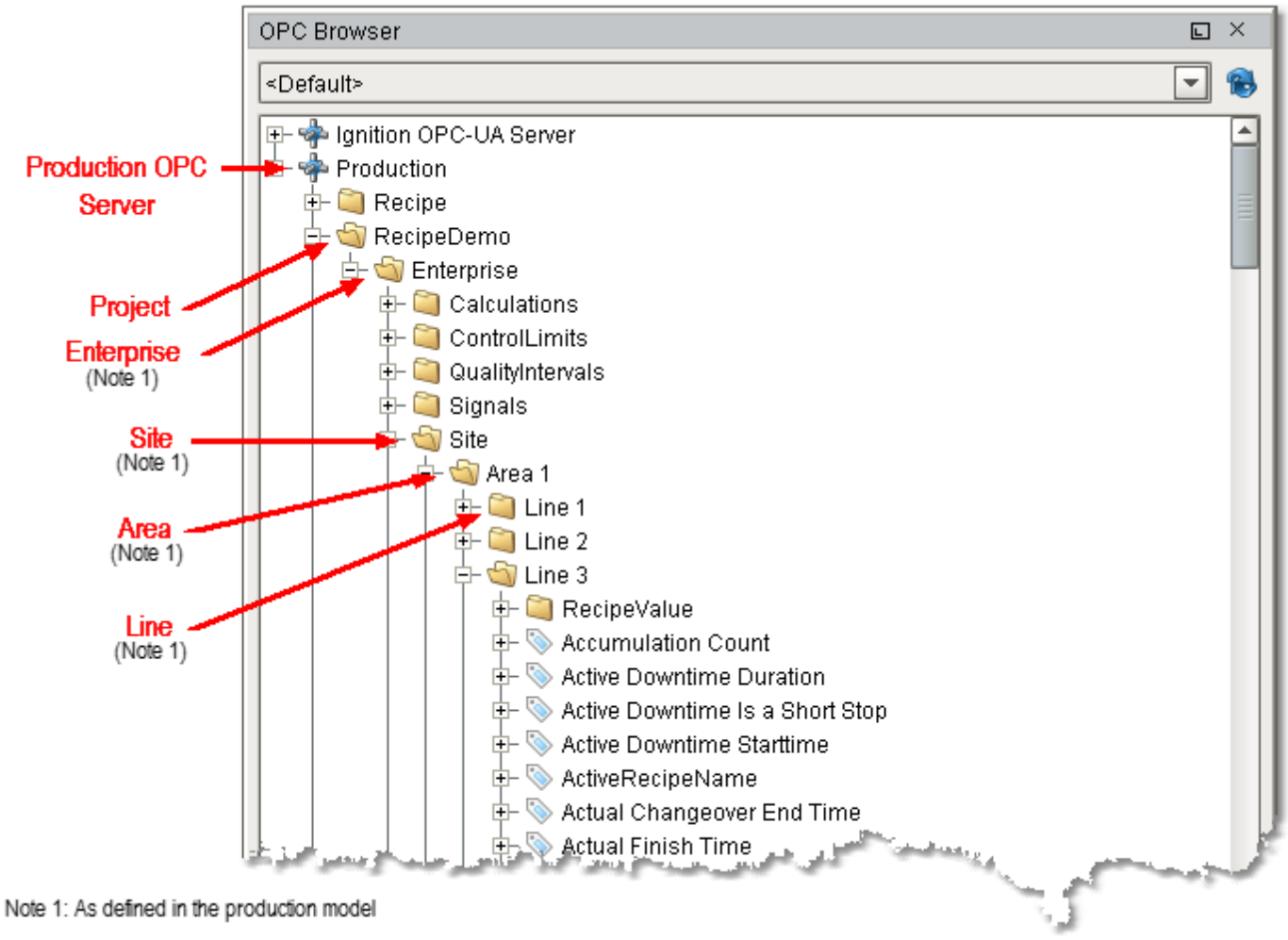
4.1.12 Production OPC Server

Production OPC Values

The production model is defined in the Ignition designer and contains your production lines, cells, cell groups and locations. Runtime access into configuration and current state of the production model is available through the Production OPC Server. It is added automatically when any of the Ignition MES modules are installed. When the production items are added, removed or modified, the changes will be reflected in the Production OPC Server when the project is saved and published in the designer.

Below are some of the values available to read, and in some cases write to for the RecipeDemo project.

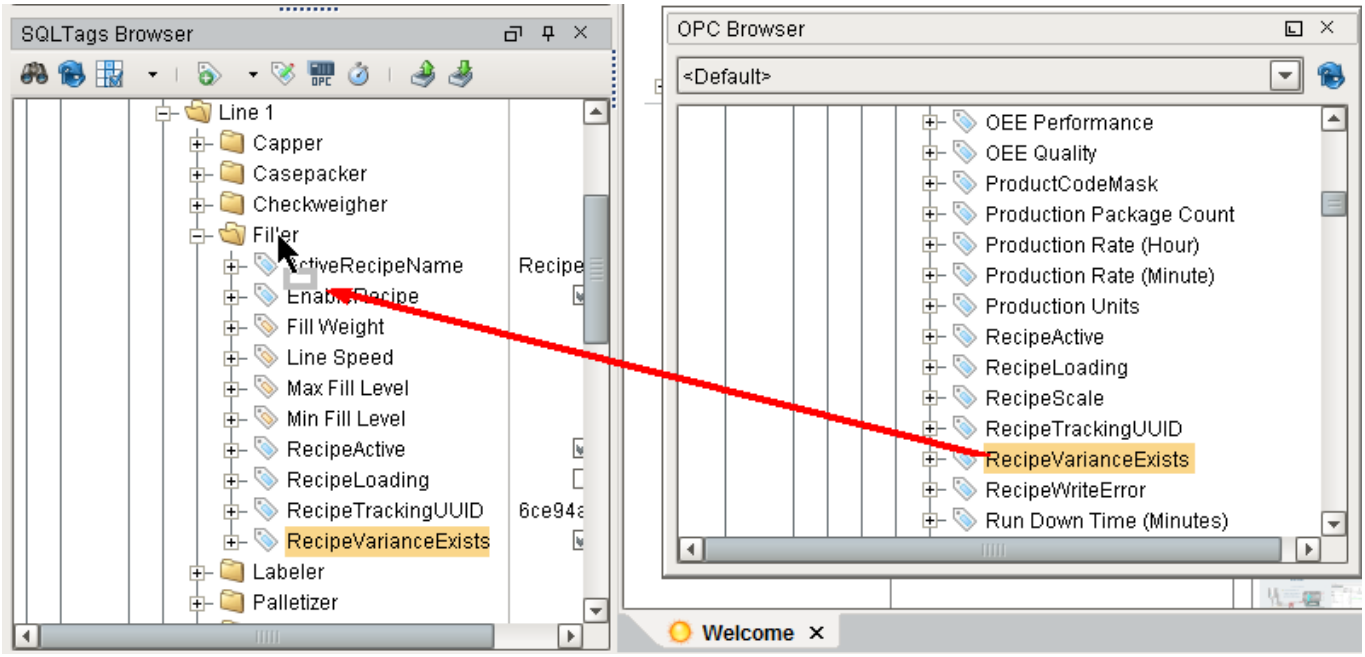
© Inductive Automation



Production Model OPC Values

To use the production OPC values in your projects, Ignition tags have to be created. The easiest method to do this is drag the production OPC value to the SQLTag Browser. Once the tag has been created, it can be used to display status on screens, used in expression and any of the other tasks that can be done with any other Ignition tags. Most of the production OPC values are read only. For example, the RecipeVarianceExists value is determined by the live production values compared to the values in the recipe (see Variance Monitoring for more details about the RecipeVarianceExists value). Because it is reflecting a status, it cannot be written to. However, others do allow writing a value to them.

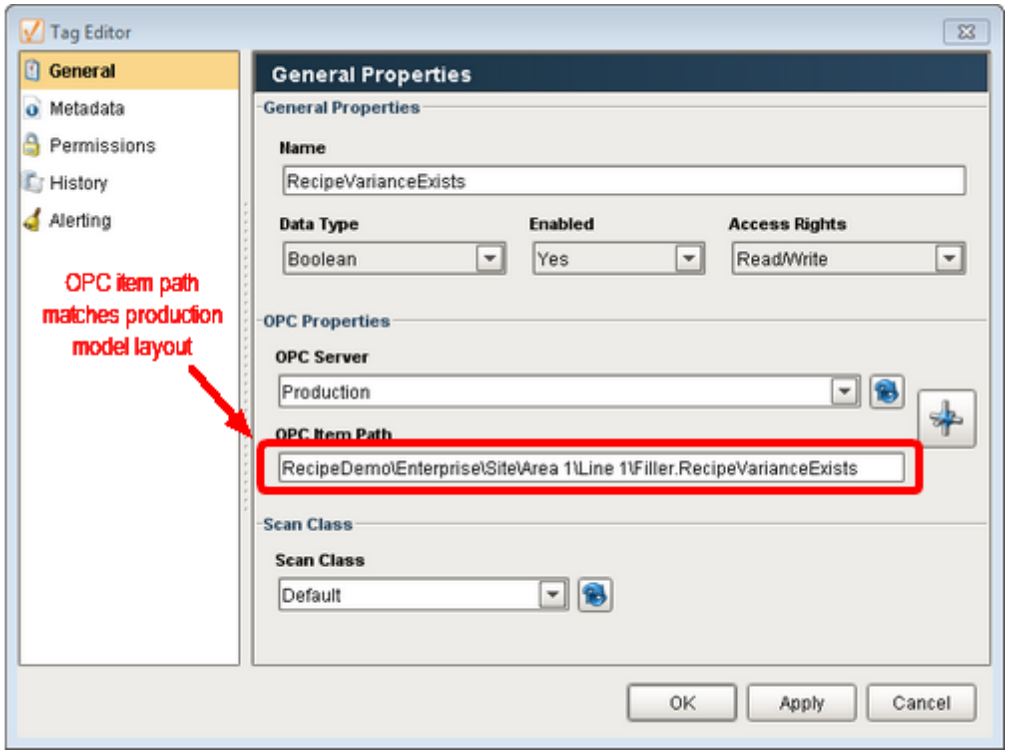
© Inductive Automation



Create Tag from Production OPC Value

Important Note

It is extremely important to understand production OPC values have an OPC item path that matches the layout of the production model. In the image below, the RecipeVarianceExists tag is shown and includes the OPC Item Path of ***RecipeDemo\Enterprise\Site\Area 1\Line 1\Filler.RecipeVarianceExists***. If tags have been previously created and the names are changed in the production model, then the OPC item path will also have to change.



Tag Configuration

© Inductive Automation

For example, if the enterprise name is changed from "Enterprise" to "My Big Company", then the OPC item path for the tag named RecipeVarianceExists will have to change to ***RecipeDemo*\My Big Company *Site**Area 1**Line 1**Filler.RecipeVarianceExists***. For this reason, it is recommended to first work on laying out your production model and make sure the names of each of the production items are what you want before creating Ignition tags.

Important Note

When writing to production OPC values that are related to production model settings, the new value is not retained upon restarting. This is because production model settings are saved in the Ignition project and is only written to the project when done so in the designer.

4.2 Installation

To install the SPC module into an existing Ignition system, follow the instructions in the Existing Ignition System. If you are installing Ignition at the same time, use the instructions in the New Ignition System.

To install the Quality Demo project, follow the steps in the Demo Installation section.

4.2.1 Existing Ignition System

4.2.1.1 Installing Modules

To install the Recipe module onto an existing Ignition server, follow the steps below:

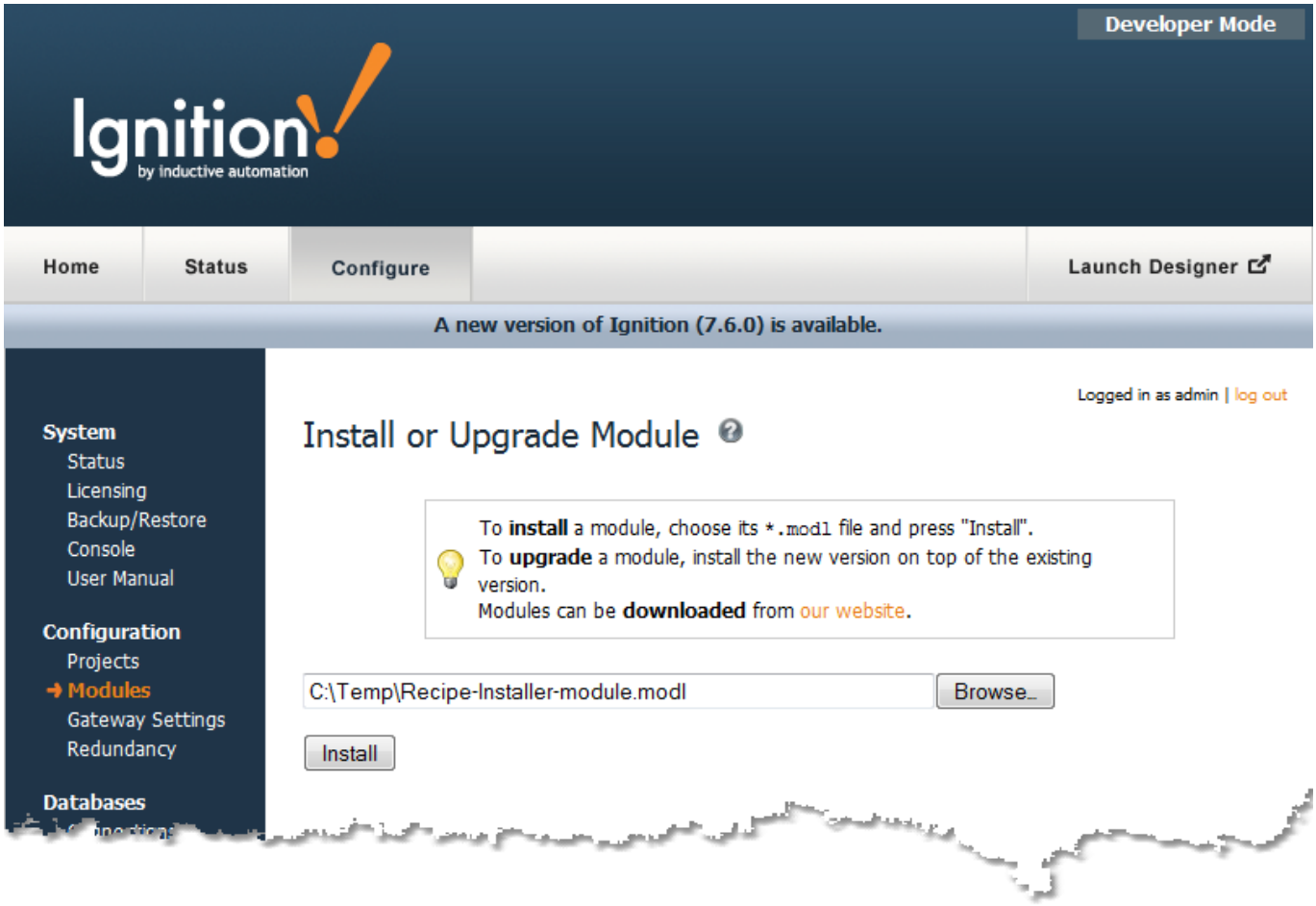
Before installing the Recipe module, it is recommended to first set up the database connection that will be used to store Recipe data.

1. **Download the Recipe-Installer-module.modl module**
   from the Inductive Automation download website. It will be under the MES modules heading.

2. **Install the Recipe-Installer-module.modl module**
   Navigate to the Modules page of the Ignition gateway. At the bottom of the list of already installed modules, click the

link. Next, browse to the Recipe-Installer-module.modl file and click the install button as shown below.

© Inductive Automation



Install Ignition Module

© Inductive Automation

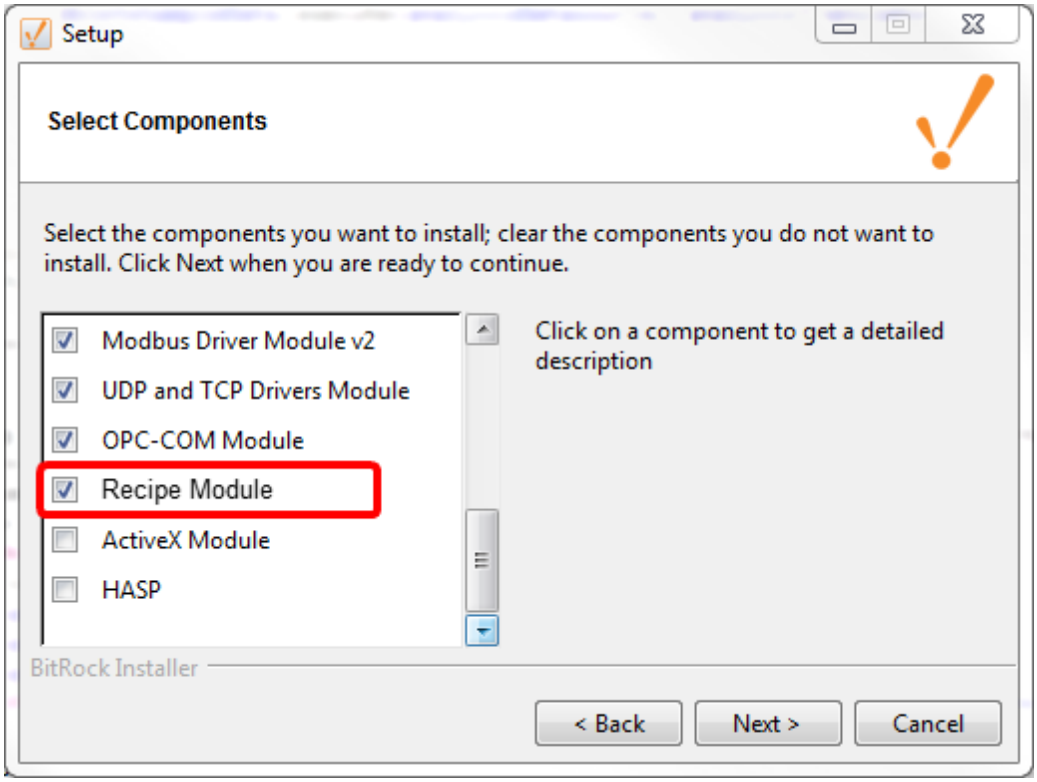
The Recipe Installer module will install all required modules. These are the Production and Recipe modules. It is important to keep in mind not to install or update these modules individually. Instead, it should be done by updating the Recipe Installer module.

4.2.2 New Ignition System

4.2.2.1 Selecting Install Options

To install the Recipe module at the same time as Ignition, add the following steps to the normal Ignition installation:

1. **Select "Custom Configuration" on the setup step during the Ignition installation.**
   The following screen will appear. Scroll down to Recipe Module and select it. This will cause the modules required for recipe functionality to be installed at the same time as Ignition.

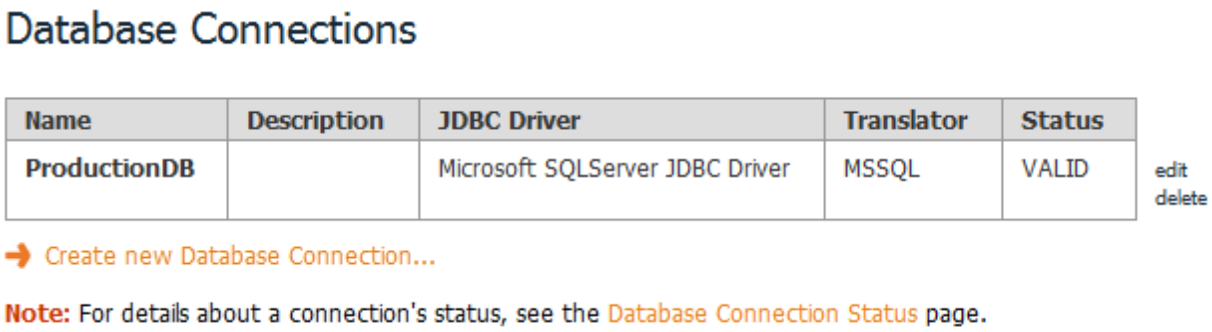

Setup
Select Components
Select the components you want to install; clear the components you do not want to install. Click Next when you are ready to continue.
Modbus Driver Module v2
UDP and TCP Drivers Module
OPC-COM Module
Recipe Module
ActiveX Module
HASP
Click on a component to get a detailed description
BitRock Installer
< Back
Next >
Cancel

Ignition Installer

4.2.3 Configure Database

Recipe data is stored in databases external to Ignition. These database(s) are setup in the gateway configuration section by selecting the **Databases> Connections** section from the left-hand configuration menu. See the Ignition documentation for more information on setting up a database connection. Below shows a typical database connection that is required for the Recipe module.

© Inductive Automation



Sample Database Connection

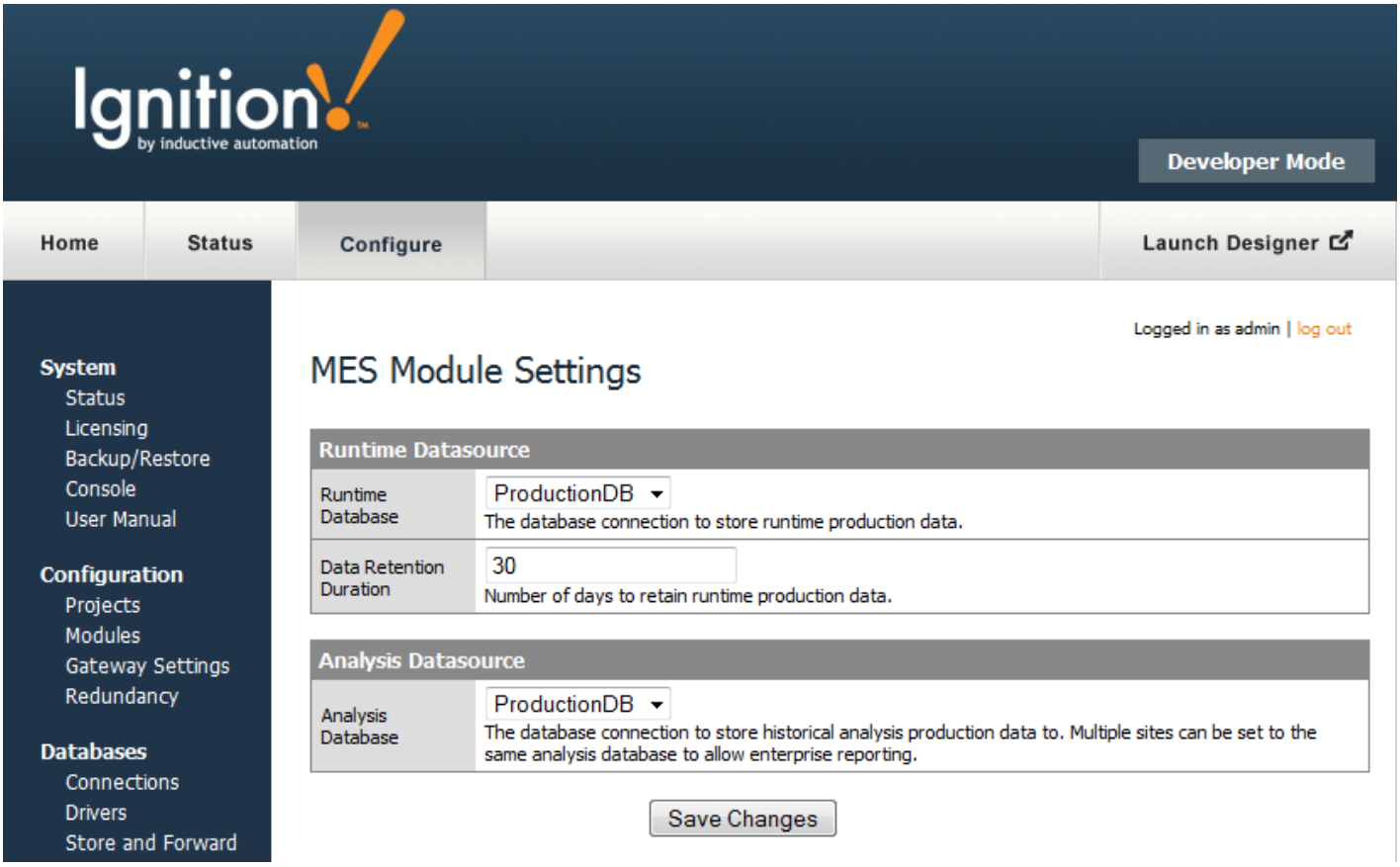
4.2.4 MES Module Settings

The OEE Downtime, Scheduling, SPC and Recipe modules store data in an SQL database. Because Ignition can be configured to multiple databases, the MES Module Settings configuration page is used to select which databases to store OEE, downtime, scheduling, SPC and Recipe data. If only one database has been configured in Ignition, then it will be selected by default.

To change the MES module settings, go to the configuration section in the gateway and select the **MES Modules> Settings** section from the left-hand side configuration menu.

Once a database connection is created, and if only one database connection exists, then it will be automatically selected to be used by the MES modules.
If more than one database connection exists, then the desired database connection can be selected to be used by the MES modules as shown below.

© Inductive Automation



MES Module Settings Page

4.3 Configuration

There are two areas to configure the Recipe module. The first area is in the Ignition Gateway and affects all MES Modules.

The second is in the Ignition Designer and is used to configure production models, user screens and the like. These settings are saved in an Ignition project and can be backed up and restored using the built-in project backup and restore features of Ignition.

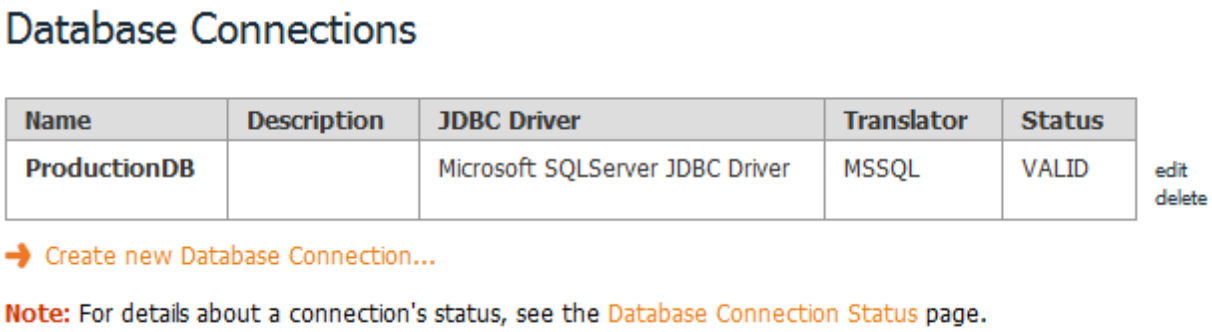
4.3.1 MES Module Configuration

The Recipe module is just one of the MES (Manufacturing Execution System) modules that has settings which can be set.

4.3.1.1 Datasource Settings

Recipe data is stored in databases external to Ignition. These database(s) are setup in the gateway configuration section by selecting the **Databases> Connections** section from the left-hand configuration menu. See the Ignition documentation for more information on setting up a database connection. Below shows a typical database connection that is required for the Recipe module.

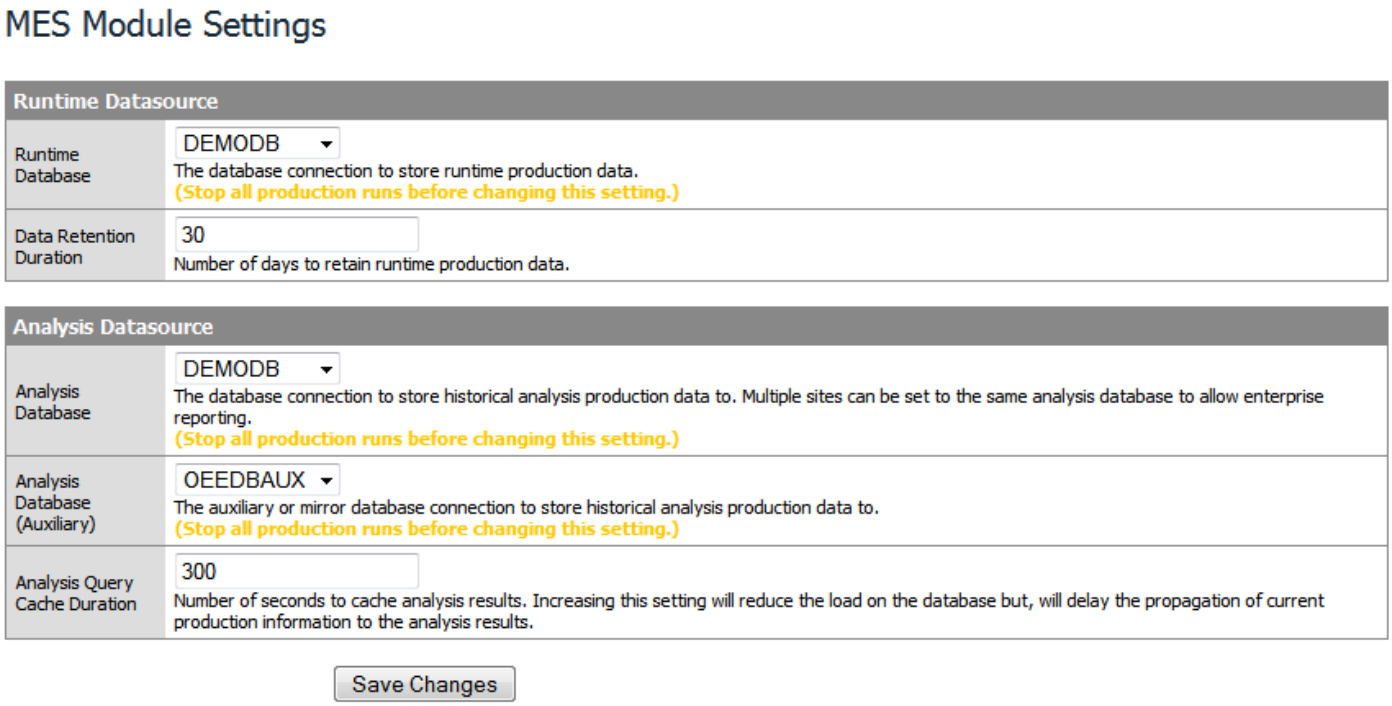
© Inductive Automation



Sample Database Connection

To change the MES module settings, go to the configuration section in the gateway and select the **MES Modules> Settings** section from the left-hand side configuration menu.

Once a database connection is created, and if only one database connection exists, then it will be automatically selected to be used by the MES modules.
If more than one database connection exists, then the desired database connection can be selected to be used by the MES modules as shown below.


MES Module Settings
Runtime Datasource
Runtime Database
DEMODB
The database connection to store runtime production data.
(Stop all production runs before changing this setting.)
Data Retention Duration
30
Number of days to retain runtime production data.
Analysis Datasource
Analysis Database
DEMODB
The database connection to store historical analysis production data to. Multiple sites can be set to the same analysis database to allow enterprise reporting.
(Stop all production runs before changing this setting.)
Analysis Database (Auxiliary)
OEEDBAUX
The auxiliary or mirror database connection to store historical analysis production data to.
(Stop all production runs before changing this setting.)
Analysis Query Cache Duration
300
Number of seconds to cache analysis results. Increasing this setting will reduce the load on the database but, will delay the propagation of current production information to the analysis results.
Save Changes

MES Module Settings Page

Runtime Database

The runtime database is where recipe data is stored.

Data Retention Duration

This setting specifies the number of days to retain the data in the runtime database after a production run has completed and does not apply to the recipe module. Recipes are kept until they are manually removed by a user.

© Inductive Automation

Analysis Database
The analysis database is where recipe change log and variances are kept.

Analysis Database (Auxiliary)
The MES Modules will mirror the historical analysis data that is written to the local analysis database to this database. For single site implementations, set this to "-none-". For multi-site implementations, set this to the datasource for the common remote enterprise database.

Analysis Query Cache Duration

This setting represents the number of seconds to cache analysis results. Analysis is used to compare recipes, view recipe change logs or view recipe variance logs.

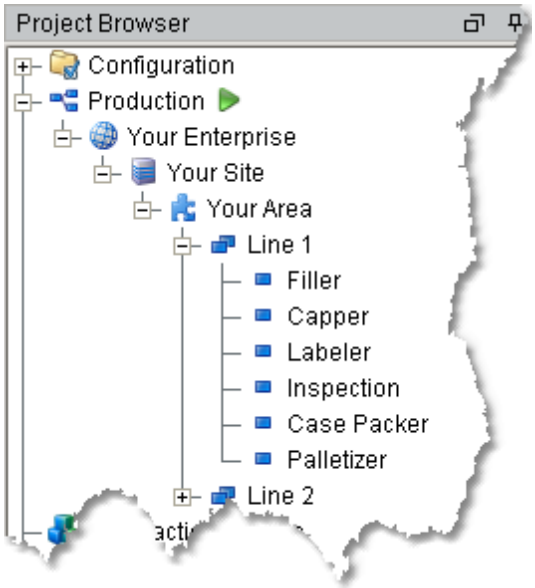
4.3.2 Production Model Configuration

A production model defines your manufacturing or process in tree view form. It provides an organized way to easily configure, control and analyze your facility. It starts with your *enterprise*, which represents your company, and continues down to the *site* (physical location), *area*, *line* and *cells*.

Recipe values can be added to lines, cell, cell groups or locations. If you are using the OEE Downtime module and have already purchased cells, you can add recipe values to them. Likewise, if you are using the SPC module and have already purchased locations, you can add recipe values to them. If you are using only using the Recipe module, then using locations is much more cost effective then using cells.

4.3.2.1 Production Model

The production model is configured within the Ignition designer and is accessed by selecting the "Production" folder in the project browser. From here your *enterprise*, *site*, *area(s)*, *line(s)* and *cell(s)* can be added, renamed and deleted.



Production Model Tree

4.3.2.1.1 Enterprise Configuration

Adding an Enterprise

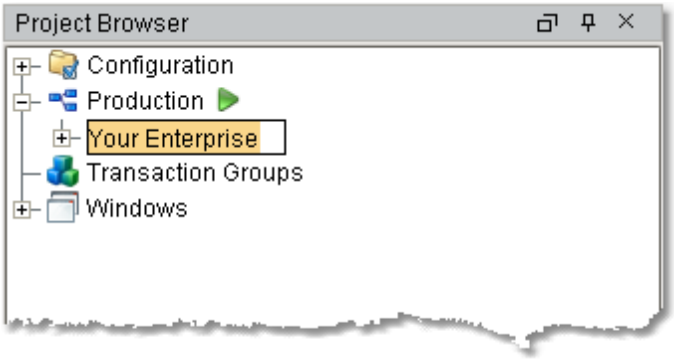
To add your *enterprise*, right-click on the "Production" folder in the project browser and select the **New Production Item > New Production Enterprise** menu item. An *enterprise* named "New Enterprise" will be added to the "Production" folder.

Renaming an Enterprise

To rename it to the name of your *enterprise*, right-click on it and select **Rename**, then enter the new name.

Important Note

It is extremely important to understand that production OPC values have an OPC item path that matches the layout of the production model and that renaming production items can cause Ignition tags associated with a production item to stop being updated. See Production OPC Server for more information.



Enterprise Name

Deleting an Enterprise

To remove an existing *enterprise*, right-click on the *enterprise* item and select the **Delete** menu item. A window will appear confirming that you permanently want to delete the production *enterprise*. Please note that the *site*, *area(s)*, *line(s), cell(s), cell group(s) and location(s)* underneath the *enterprise* will also be permanently removed.

© Inductive Automation

General Enterprise Settings

For the *enterprise*, there are only general settings. These settings are accessed by selecting the *enterprise* item contained in the ***"Production"*** folder in the project browser and then selecting the "General" tab as shown below.

Enterprise General Settings

Enabled — By default, added *enterprises* are enabled. It can be disabled by un-checking the Enabled setting and saving the project. This will stop the OEE, downtime and scheduling module from executing the *enterprise*, the *site* and all *area(s)*, *line(s)* and *cell(s)* that are underneath it.

Description — This is an optional description and is just for your reference.

Recipe Values

Recipe values can be added to enterprises but, tags cannot be associated with them. The recipe values that are added to an enterprise are propagated down to production sites, areas, lines, cells, cell groups and location beneath the enterprsie. This provides a quick method to add recipe values that are common to all machines within an enterprise. It also allows for the ability to propagate a value of a recipe value down to all production items beneath the enterprise. See Production Model for more information on propagating recipe values to child production items. See Recipe Types for more information about recipe values and Recipe Values for more information about configuring recipe values.

4.3.2.1.2 Site Configuration

Adding a Site

To add your *site*, right-click on your *enterprise* folder in the project browser and select the **New Production Item > New Production Site** menu item. A *site* named "New Site" will be added to the *enterprise* folder.

Renaming a Site

To rename it to the name representing the *site's* physical location, right-click on it and select **Rename**, then enter the new name.

Important Note

It is extremely important to understand production OPC values have an OPC item path that matches the layout of the production model and that renaming production items can cause Ignition tags associated with a production item to stop being updated. See Production OPC Server for more information.

Deleting a Site

To remove an existing *site*, right-click on the *site* item and select the **Delete** menu item. A window will appear confirming that you permanently want to delete the production *site*. Please note that the *area(s)*, *line(s),* cell(s), cell group(s) and location(s) underneath the *site* will also be permanently removed.

New Site

General Site Settings

These settings are accessed by selecting the *site* item contained in the *enterprise* folder in the project browser, and then selecting the "General" tab.

Enabled — By default, added *sites* are enabled. It can be disabled by un-checking the Enabled setting and saving the project. This will stop the MES modules from executing the *site* and all *area(s)*, *line(s), cell(s), cell group(s) and location(s)* that are underneath it.

Description — This is an optional description and is just for your reference.

Recipe Values

Recipe values can be added to sites, but tags cannot be associated with them. The recipe values that are added to a site are propagated down to production areas, lines, cells, cell groups and location beneath the site. This provides a quick method to add recipe values that are common to all machines beneath a site. It also allows for the ability to propagate a value of a recipe value down to all production items beneath the site. See Production Model for more information on propagating recipe values to child production items. See Recipe Types for more information about recipe values and Recipe Values for more information about configuring recipe values.

4.3.2.1.3 Area Configuration

Adding an Area

To add a production *area*, right-click on your *site* folder in the project browser and select the **New Production Item > New Production Area** menu item. An *area* named "New Area" will be added to the *site* folder. Multiple production *areas* can be added to your production *site*. Each *area* can represent a physical or logical production *area* within your production *site*. Some examples of production *areas* are: packaging, cracking, filtration, fabrication, etc.

Renaming an Area

To rename it to the name representing the production *area*, right-click on it and select **Rename**, then enter the new name.

Important Note

It is extremely important to understand production OPC values have an OPC item path that matches the layout of the production model and that renaming production items can cause Ignition tags

associated with a production item to stop being updated. See Production OPC Server for more information.

Deleting an Area

To remove an existing production *area*, right-click on the *area* item and select the **Delete** menu item. A window will appear confirming that you permanently want to delete the production *area*. Please note that the *line(s), cell(s), cell group(s) and location(s)* underneath the *area* will also be permanently removed.

New Area

Area General Settings

These settings are accessed by selecting the desired *area* item contained in the *site* folder in the project browser and then selecting the "General" tab.

Enabled By default, added *areas* are enabled. It can be disabled by un-checking the Enabled setting and saving the project. This will stop the MES modules from executing the *area* and all *line(s), cell(s), cell group(s) and location(s)* that are underneath it.

Description This is an optional description and is just for your reference.

Recipe Values

Recipe values can be added to areas but, tags cannot be associated with them. The recipe values that are added to an area are propagated down to production lines, cells, cell groups and location beneath the area. This provides a quick method to add recipe values that are common to all machines beneath an area. It also allows for the ability to propagate a value of a recipe value down to all production items beneath the area. See Production Model for more information on propagating recipe values to child production items. See Recipe Types for more information about recipe values and Recipe Values for more information about configuring recipe values.

4.3.2.1.4 Line Configuration

Adding a Line

To add a production *line*, right-click on an *area* folder in the project browser and select the **New Production Item > New Production Line** menu item. A *line* named "New Line" will be added to the *area* folder. Multiple production *lines* can be added to a production *area*.

Renaming a Line

To rename it to the name representing the production *line*, right-click on it and select **Rename**, then enter the new name.

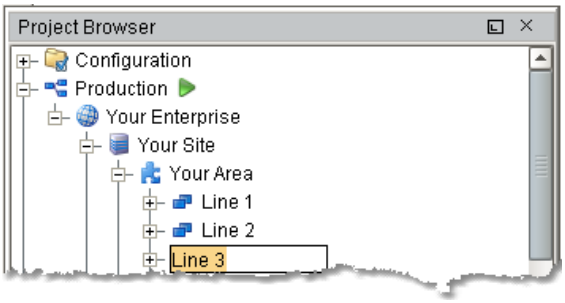
Important Note

It is extremely important to understand production OPC values have an OPC item path that matches the layout of the production model and that renaming production items can cause Ignition tags

associated with a production item to stop being updated. See Production OPC Server for more information.

Deleting a Line

To remove an existing production *line*, right-click on the *line* item and select the **Delete** menu item. A window will appear confirming that you permanently want to delete the production *line*. Please note that the cell(s), cell group(s) and location(s) underneath the *line* will also be permanently removed.



New Line

Line General Settings

These settings are accessed by selecting the desired *line* item contained in the *area* folder in the project browser and then selecting the "General" tab.

Enabled By default, added *lines* are enabled. It can be disabled by un-checking the Enabled setting and saving the project. This will stop the MES modules from executing the *line* and *cell(s), cell group(s) and location(s)* that are underneath it.

Description This is an optional description and is just for your reference.

Sub Recipe Mask

This Sub Recipe Mask is required when using the sub recipe feature. If the sub recipe feature is not being used for the production item, leave it blank. See Sub Recipes for more information above sub recipes and Sub Recipe Mask for how to use this setting.

Recipe Values

Recipe values can be added to lines and can represent settings that are written to a PLC or other controller associated with a machine or be used internally in Ignition.
The recipe values that are added to a line are propagated down to production cells, cell groups and location beneath the line. This provides a quick method to add recipe values that are common to all machines beneath a line. It also allows for the ability to propagate a value of a recipe value down to all production items beneath the area. See Production Model for more information on propagating recipe values to child production items. See Recipe Types for more information about recipe values and Recipe Values for more information about configuring recipe values.

© Inductive Automation

4.3.2.1.5 Cell Configuration

Adding a Cell

To add a production cell, right-click on a *line* folder in the project browser and select the **New Production Item > New Production Cell** menu item. A cell named "New Cell" will be added to the *line* folder. Multiple production cells can be added to a production *line*.

Renaming a Cell

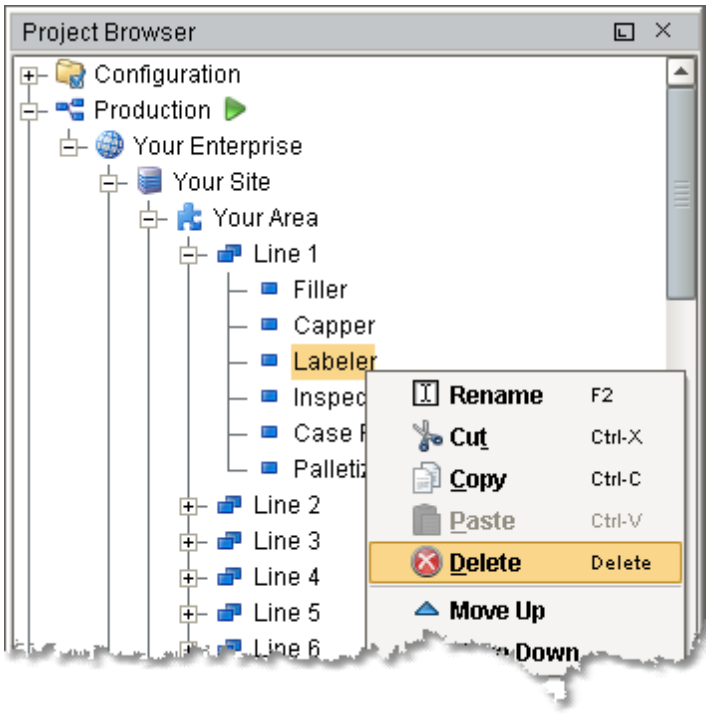
To rename it to the name representing the production cell, right-click on it and select **Rename**, then enter the new name.

Important Note

It is extremely important to understand production OPC values have an OPC item path that matches the layout of the production model and that renaming production items can cause Ignition tags associated with a production item to stop being updated. See Production OPC Server for more information.

Deleting a Cell

To remove an existing production cell, right-click on the cell item and select the **Delete** menu item. A window will appear confirming that you permanently want to delete the production cell.



Deleting a Cell

Cell General Settings

These settings are accessed by selecting the desired cell item contained in the *line* folder in the project browser and then selecting the "General" tab.

Enabled By default, added cells are enabled. It can be disabled by un-checking the Enabled setting and saving the project. This will stop the MES modules from executing the cell.

Description This is an optional description and is just for your reference.

Sub Recipe Mask

This Sub Recipe Mask is required when using the sub recipe feature. If the sub recipe feature is not being used for the production item, leave it blank. See Sub Recipes for more information above sub recipes and Sub Recipe Mask for how to use this setting.

Recipe Values

Recipe values can be added to cells and can represent settings that are written to a PLC or other controller associated with a machine or be used internally in Ignition. See Recipe Types for more information about recipe values and Recipe Values for more information about configuring recipe values.

4.3.2.1.6 Cell Group Configuration

Adding a Cell Group

To add a production cell group, right-click on a *line* folder in the project browser and select the **New Production Item > New Production Cell Group** menu item. A cell group named "New Cell Group" will be added to the *line* folder. Multiple production cell groups can be added to a production *line*.

Renaming a Cell Group

To rename it to the name representing the production cell group, right-click on it and select **Rename**, then enter the new name.

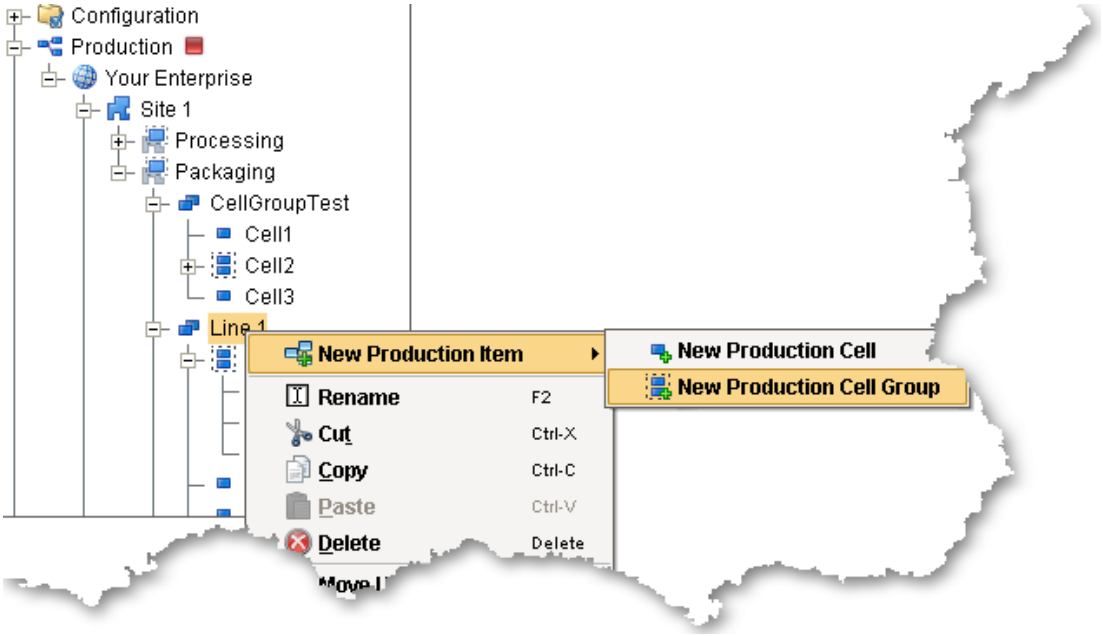
Important Note

It is extremely important to understand production OPC values have an OPC item path that matches the layout of the production model and that renaming production items can cause Ignition tags associated with a production item to stop being updated. See Production OPC Server for more information.

Deleting a Cell Group

To remove an existing production cell group, right-click on the cell group item and select the **Delete** menu item. A window will appear confirming that you permanently want to delete the production cell group. Please note that the cell(s) underneath the cell group will also be permanently removed.

© Inductive Automation



Adding a Cell Group

Cell Group General Settings

These settings are accessed by selecting the desired cell group item contained in the *line* folder in the project browser and then selecting the "General" tab.

Enabled By default, added cell groups are enabled. It can be disabled by un-checking the Enabled setting and saving the project. This will stop the OEE, downtime and scheduling module from executing the cell group.

Description This is an optional description and is just for your reference.

Sub Recipe Mask

This Sub Recipe Mask is required when using the sub recipe feature. If the sub recipe feature is not being used for the production item, leave it blank. See Sub Recipes for more information above sub recipes and Sub Recipe Mask for how to use this setting.

Recipe Values

Recipe values can be added to cell groups and can represent settings that are written to a PLC or other controller associated with the cell group or be used internally in Ignition.
The recipe values that are added to a cell group are propagated down to production cells beneath the cell group. This provides a quick method to add recipe values that are common to all machines beneath a cell group. It also allows for the ability to propagate a value of a recipe value down to all production items beneath the cell group. See Production Model for more information on propagating recipe values to child production items. See Recipe Types for more information about recipe values and Recipe Values for more information about configuring recipe values.

© Inductive Automation

4.3.2.1.7 Location Configuration

Adding a Location

To add a production cell, right-click on a *line* folder in the project browser and select the **New Production Item > New Production Cell** menu item. A cell named "New Cell" will be added to the *line* folder. Multiple production cells can be added to a production *line*.

Renaming a Location

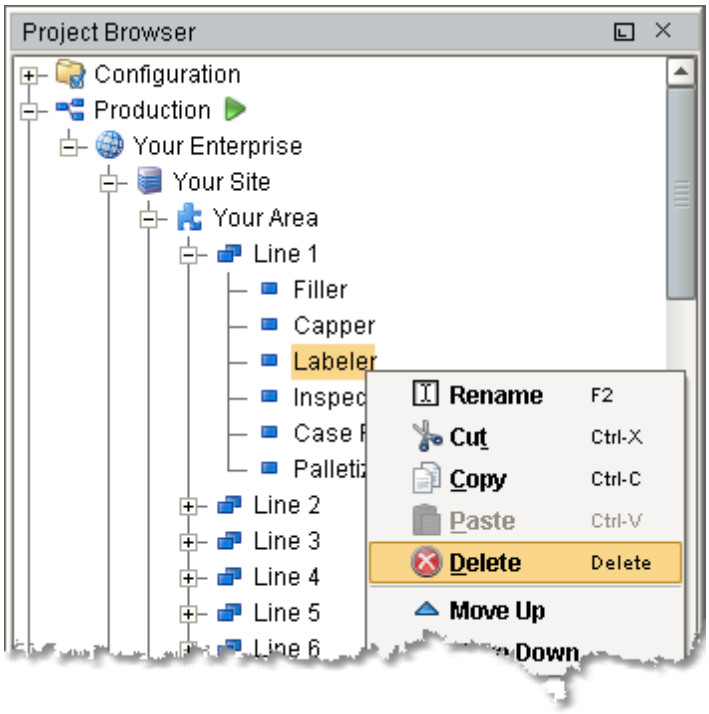
To rename it to the name representing the production cell, right-click on it and select **Rename**, then enter the new name.

Important Note

It is extremely important to understand production OPC values have an OPC item path that matches the layout of the production model and that renaming production items can cause Ignition tags associated with a production item to stop being updated. See Production OPC Server for more information.

Deleting a Location

To remove an existing production cell, right-click on the cell item and select the **Delete** menu item. A window will appear confirming that you permanently want to delete the production cell.



Deleting a Cell

Location General Settings

These settings are accessed by selecting the desired cell item contained in the *line* folder in the project browser and then selecting the "General" tab.

Enabled By default, added cells are enabled. It can be disabled by un-checking the Enabled setting and saving the project. This will stop the OEE, downtime and scheduling module from executing the cell.

© Inductive Automation

Description This is an optional description and is just for your reference.

Sub Recipe Mask
This Sub Recipe Mask is required when using the sub recipe feature. If the sub recipe feature is not being used for the production item, leave it blank. See Sub Recipes for more information above sub recipes and Sub Recipe Mask for how to use this setting.

Recipe Values
Recipe values can be added to locations and can represent settings that are written to a PLC or other controller associated with a machine or be used internally in Ignition. See Recipe Types for more information about recipe values and Recipe Values for more information about configuring recipe values.

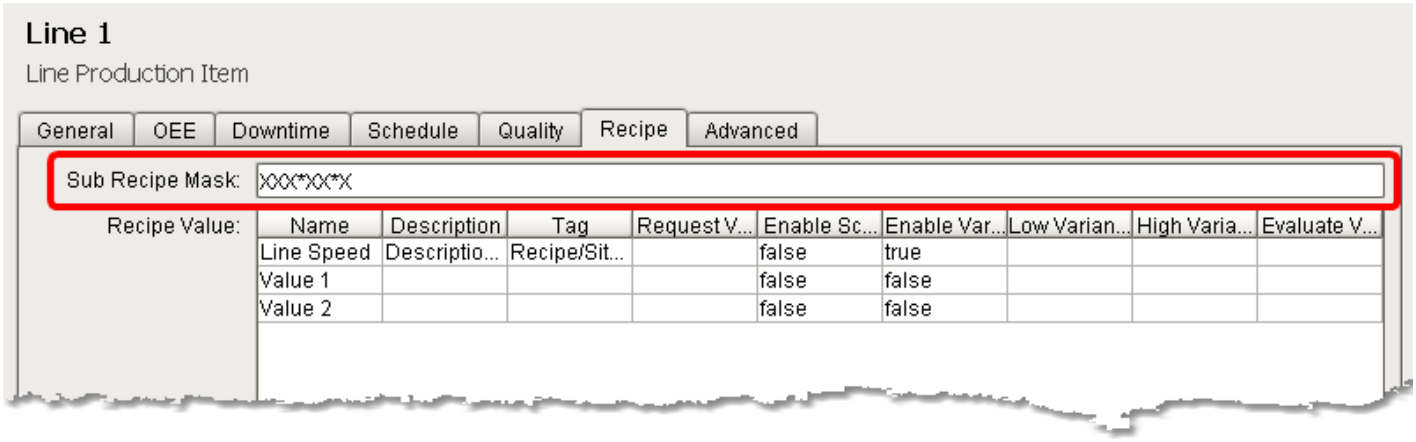
4.3.3 Sub Recipe Mask

This Sub Recipe Mask is required when using the sub recipe feature and only apply to line, cell, cell group or location type of production items. If the sub recipe feature is not being used for the production item, leave it blank. The sub recipe feature can be used on a production item by production item basis. This means that standard recipe and sub recipe functionality can be mixed. For example, you may have a tape machine that recipe values only change based on the case size. If there are only two different case sizes and there is a digit in the product code that specifies the case size, then sub recipes can be used for the tape machine. All other machines can use the normal recipe functionality.

4.3.3.1 Sub Recipe Mask Setting

In some situations, a limited number of digits in the product code can specify the recipe to use for a machine. This is accomplished in the Recipe / Changeover Module by setting a sub recipe mask. See Sub Recipes for more information.

To set the Sub Recipe Mask for a line, cell, cell group or location, first select the desired production item. Next, select the Recipe tab and enter the new Sub Recipe Mask value.



Sub Recipe Mask Setting

4.3.4 Recipe Values

Recipe values are defined by production item. Each machine, process or other equipment will have settings that are unique. For example, a casepacker will not have the same settings as a mixer, so this is why recipe values are defined by production line, cell, cell group or location.

© Inductive Automation

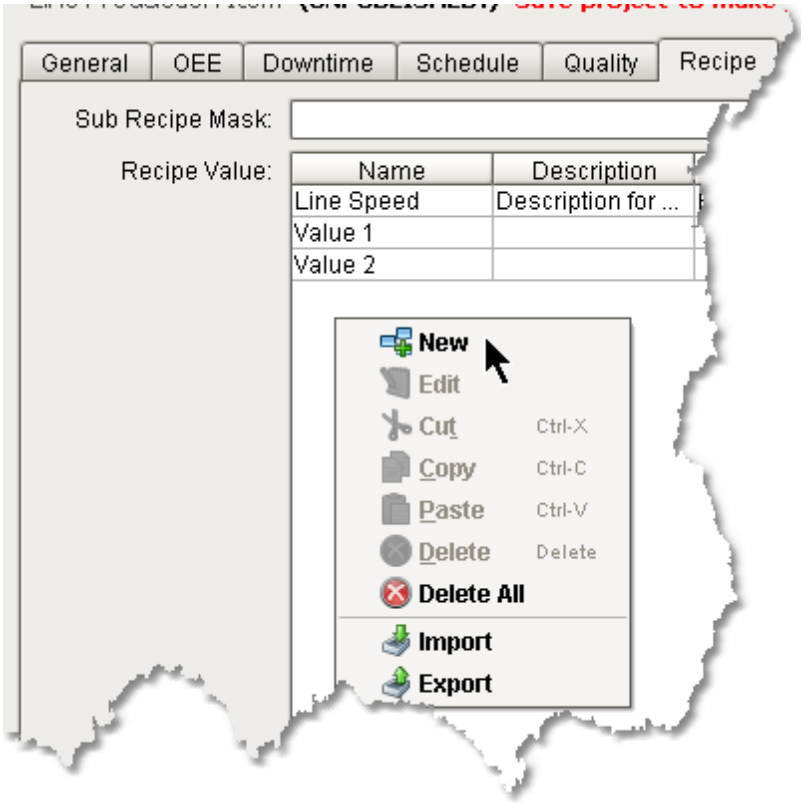
See Recipe Types for more information on how recipe values work. The following sections detail how to add, edit, delete, export and import recipe values for a production item.

4.3.4.1 Adding a Recipe Value

There is more than one method to add recipe values to a production line, cell (machine), cell group or location.

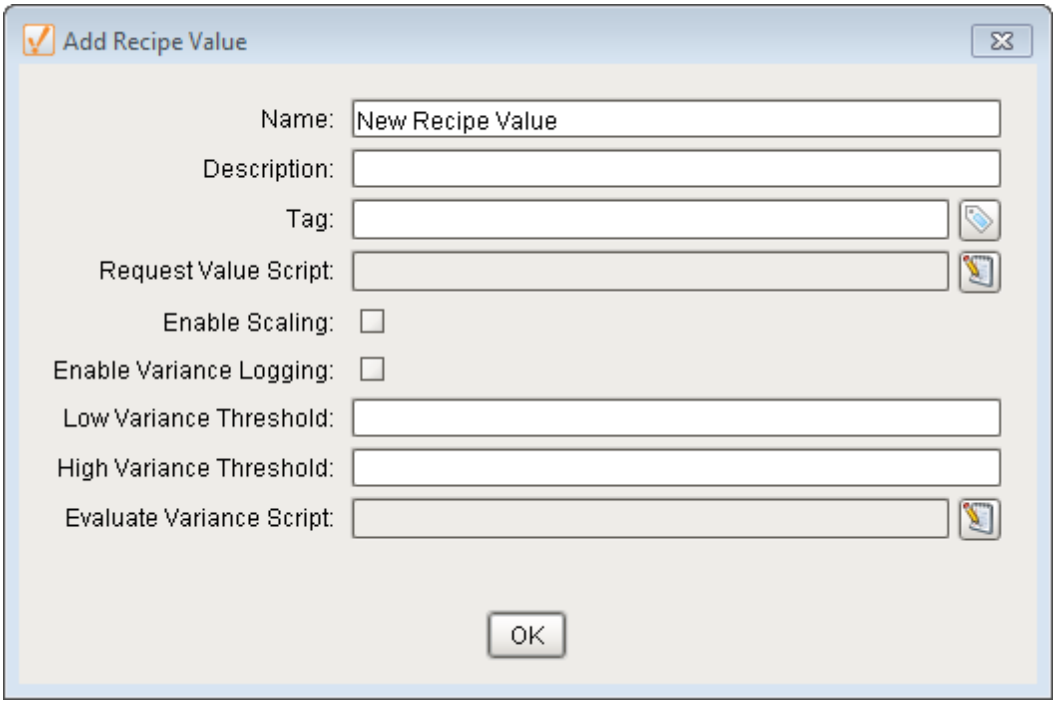
Create Recipe Value

To create a new recipe value for a production line, cell, cell group or location, first select the production item. Next, select the Recipe tab and right click in the recipe value table. A popup menu will appear as shown in the image below.



Recipe Value Table Popup Menu

Select the New menu item and the Add Recipe Value window will appear. Configure the new recipe value based on the desired functionality that is described below.



Adding a Recipe Value

Name (required)
The required name is used to reference the recipe value. The name must be unique and must not exist in any of the child production items of the production item that the recipe value is being added to. The reason for this is that recipe values are propagated down to all of the children, and if the name is the same, a conflict will occur. Also, some characters are not allowed in recipe value names.

Description
The recipe value description is used to further describe the recipe value. It appears in the recipe editor component, analysis, reports, and etc.

Tag (required)
This is the path to the Ignition tag that is associated with this recipe value. If a recipe value is added but a tag is assigned, it will not appear in the recipe editor, and values will not be used when recipes are selected.

Request Value Script
Optionally, script can be added to calculate or obtain a value to return for a recipe value anytime a recipe is selected for a production item. This provides flexibility to do just about anything in place of returning the value stored in the recipe management system. See Request Value Script section for more information.

Enable Scaling
If this option is checked, the recipe value will be scaled. The recipe value is retrieved out of the recipe management system and then scaled by the value of the recipe scale tag for the production item.

See Recipe Scaling for more information.

Enable Variance Logging

© Inductive Automation

If this option is checked, then the tag will be monitored for changes after a recipe is selected for a production item. If the value changes more than the window defined in the Low Variance Threshold and High Variance Threshold, it will be logged to the database, and the recipe variances exists tag for the production item will be set to true. This prevents values that are know to vary within an allowable range from being logged to the database and causing the recipe variances exists tag from being set.

If this option is not checked, then the value can change and it will not be logged to the database. Also, the recipe variances exists tag will not be set as a result of this recipe value.

See Variance Tracking for more information.

Low Variance Threshold

The Low Variance Threshold setting is used to define the lower limit before recipe variances are triggered for this recipe value. The variance threshold can be defined as a percentage of the recipe value or a fixed amount.

See Variance Tracking for more information.

High Variance Threshold

The High Variance Threshold setting is used to define the upper limit before recipe variances are triggered for this recipe value. The variance threshold can be defined as a percentage of the recipe value or a fixed amount.

See Variance Tracking for more information.

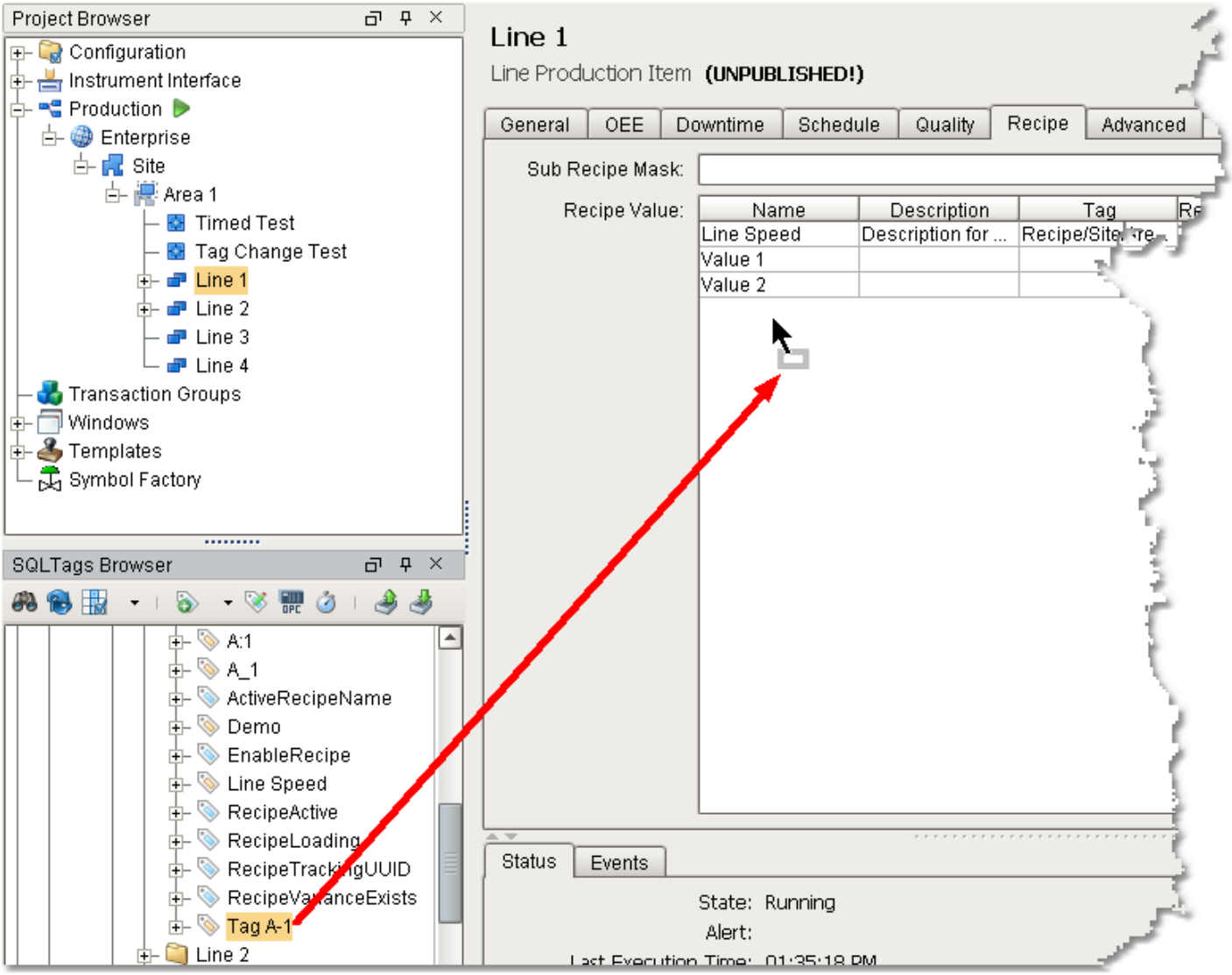
Evaluate Variance Script

Optionally, script can be used instead of using the Low Variance Threshold and High Variance Threshold settings to determine if the recipe value is outside of an allowable range. When an Ignition tag value change is detected, the variance state is evaluated using the Low Variance Threshold and High Variance Threshold settings. Then if an Evaluate Variance Script has been entered for the recipe value, the script will be executed, and the state can be changed. See Variance Tracking and Evaluate Variance Script for more information.

Create Recipe Value by Drag and Drop

Recipe values can be added to production items as described above, but the easiest method is to drag one or more tags from the SQLTags Browser to the recipe value table. Multiple tags can be selected by holding down the ctrl or shift keys while selecting tags in the SQLTags Browser and dragged to the Recipe Values table to create multiple recipe values at once. When using this method to add a one or more recipe values, drag the selected Ignition tags to the open area on the recipe value table.

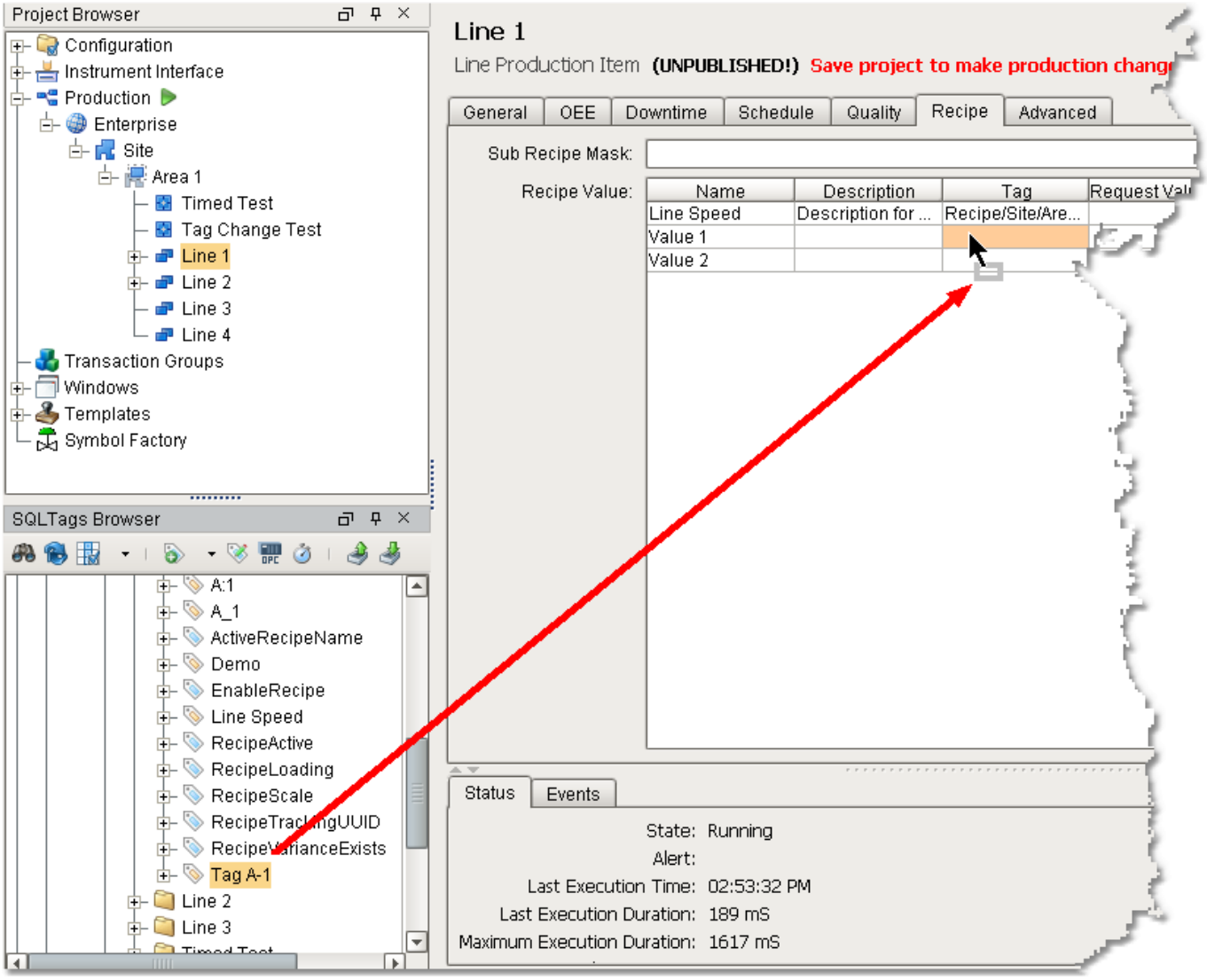
© Inductive Automation



Adding a Recipe Value Using Drag and Drop

Assigning Tags to Recipe Values

In cases where recipe values propagated down from a parent production item or recipe values have been previously created with no tag assigned to them, tags can be assigned to them using drag and drop. This can only be done one tag to recipe value at a time.



Assigning a Tag to Recipe Value Using Drag and Drop

4.3.4.2 Editing a Recipe Value

To edit a Recipe Value, select the existing Recipe Value you wish to edit, then right-click and select "Edit" from the menu. The same window used to add recipe values will appear, allowing the information to be edited.

4.3.4.3 Deleting a Recipe Value

To delete a Recipe Value, select the existing Recipe Value you wish to remove, then right-click and select "Delete" from the menu. A window will appear confirming that you permanently want to delete the Recipe Value.

4.3.4.4 Import / Export

To import recipe value configuration entries, right-click anywhere on the recipe values table and select the **Import** menu item. A dialog box will appear to allow selection of a comma separated values (csv) formatted file.

The first line of the file must at least contain the property names separated by commas. If additional names exist, they will be ignored. The property names can be in any order. Below is a sample csv file

showing multiple recipe value configuration entries.

```
ValueName,ValueDescription,ValueSQLTag,ValueCalcScript,AllowScaling,ValueMonitorEnabled,
ValueMonitorLow,ValueMonitorHigh,ValueMonitorScript
"Line Speed","Description for Line Speed","Recipe/Site/Area 1/Line 1/Line Speed","","false","true","","",""
"Value 1","","","","false","false","","",""
"Value 2","","","","false","false","","",""
```

To export recipe value configuration entries, right-click anywhere on the table containing recipe value configuration entries and select the **Export** menu item. A dialog box will appear to allow selection of an existing file or the typing in of a name of the new file to save the recipe value configuration entries to. If a file extension is not entered, then the default .csv will be used.

4.4 Component Reference

This section is a reference for all the components that come with the Recipe Module.

4.4.1 Recipe Editor

Description

A component that is added to Ignition windows to manage recipes. This is just one method of managing recipes and for more information on the other methods see the Editing Recipes section.

It has the capability for end users to do the following recipe related tasks:

- Manage recipe value security.
- Manage sub product code recipes.
- Manage default machine values.
- Manage master recipes.
- Manage machines that a recipe can be run.

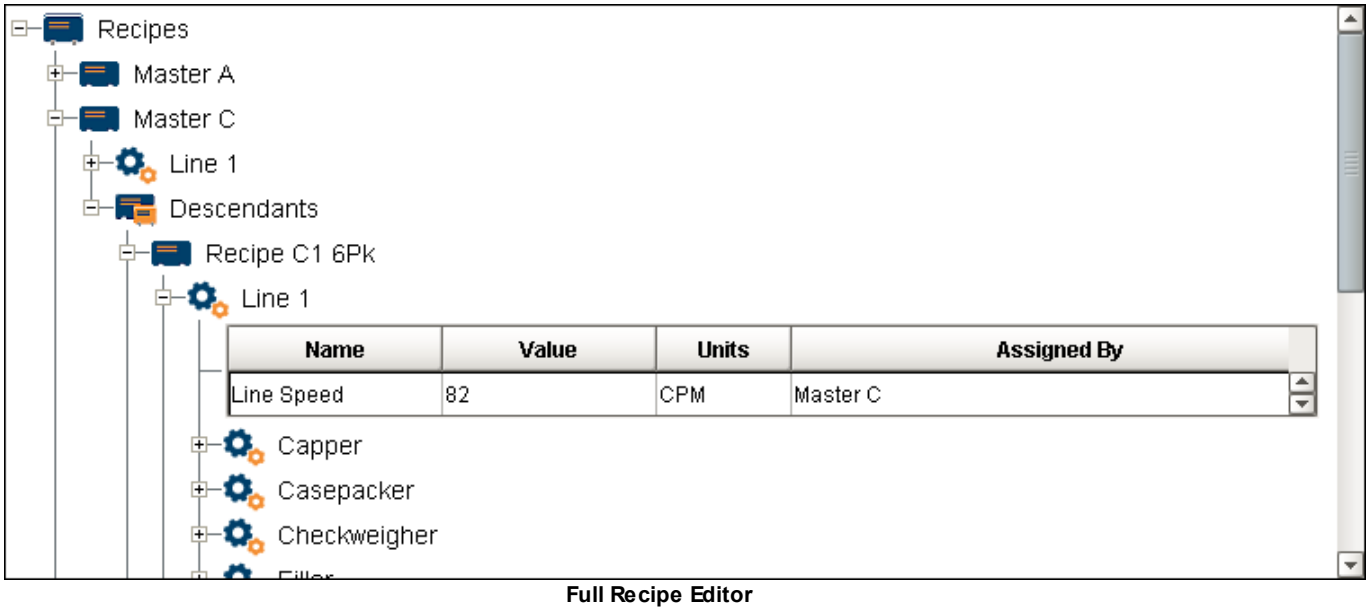
Based on the setting of the properties of the Recipe Editor component, more or less detail can be shown. This provides a method of displaying the correct amount of information depending on the logged in user's authentication roles. For example, the image below is very clean only showing limited recipes. This mode allows changing of recipe values for final recipes (not in master recipes). See Master Recipes for more information.

Basic Recipe Editor

Where as this image demonstrates a lot of recipe information with many more options, providing much more configuration of recipes. The Show Master Recipes property will determine if master recipes are shown. See Master Recipes for more information.

New master recipes can be added by right clicking on the on the root Recipes node and selecting the Add Recipe menu item.

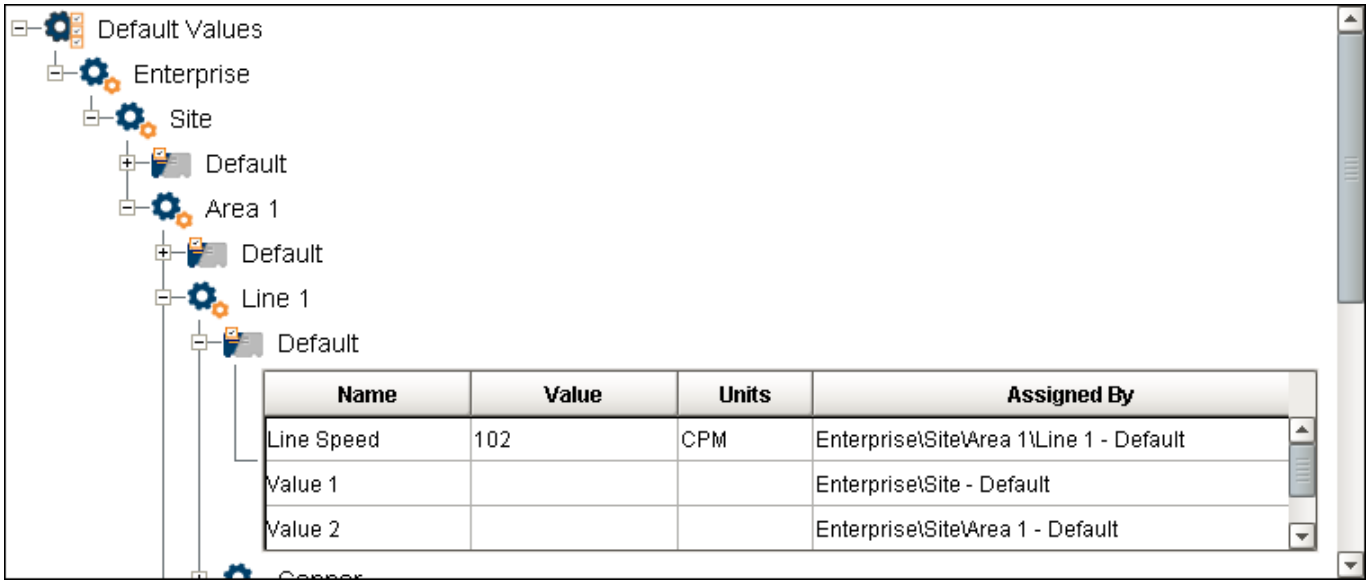
New descendant recipes can be added by right clicking on the Descendants node and selecting the Add Recipe menu item. Existing descendant recipes can be renamed, removed or etc. by right clicking on the descendant recipe node and selecting the desired menu item.



Full Recipe Editor

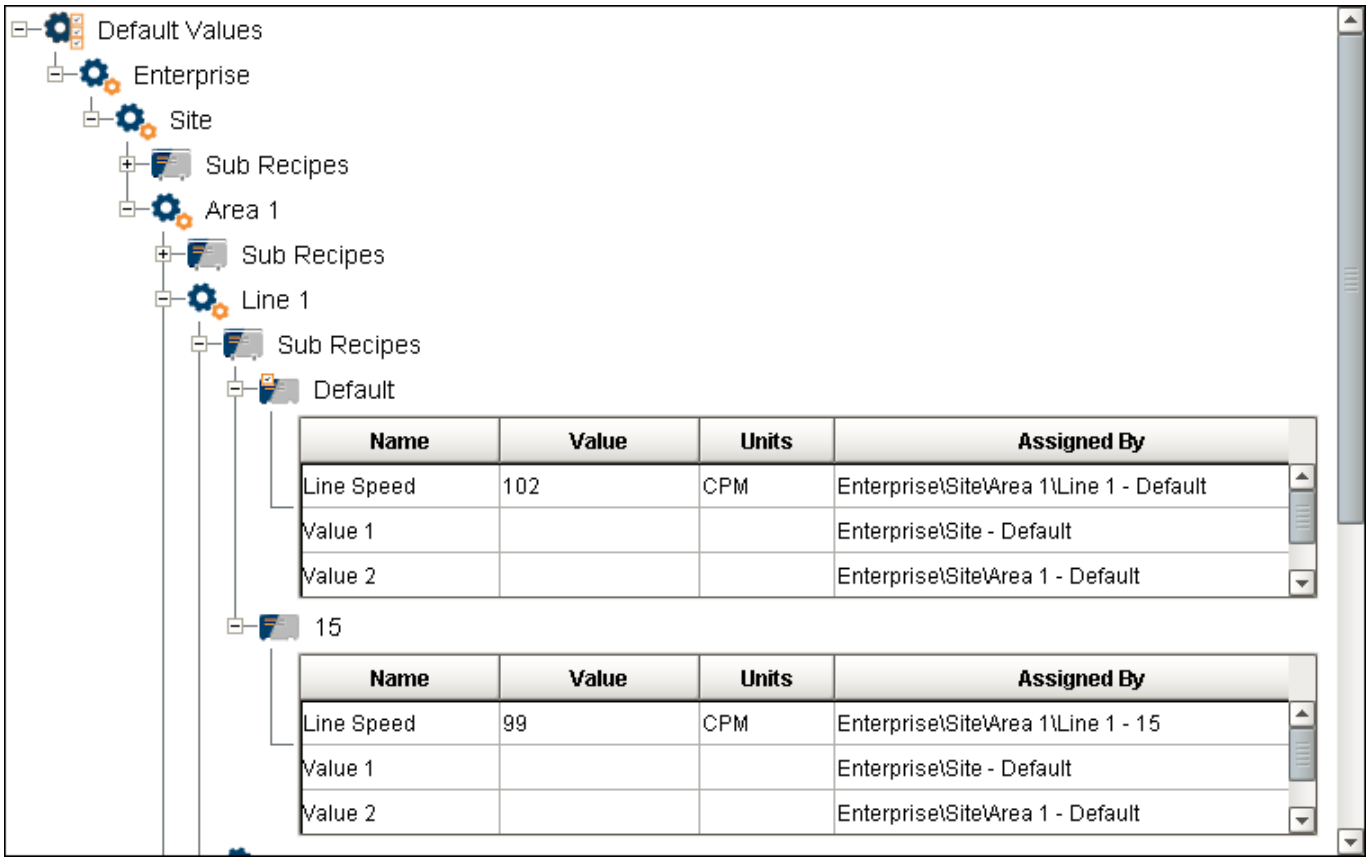
In addition to editing recipes, default values for machines (production lines, cells, cell groups and locations) can be managed as shown in the image below. The Show Item Defaults property determines if the default values root item is shown in the recipe editor. See Default Values for more information.

© Inductive Automation



Default Value Editor

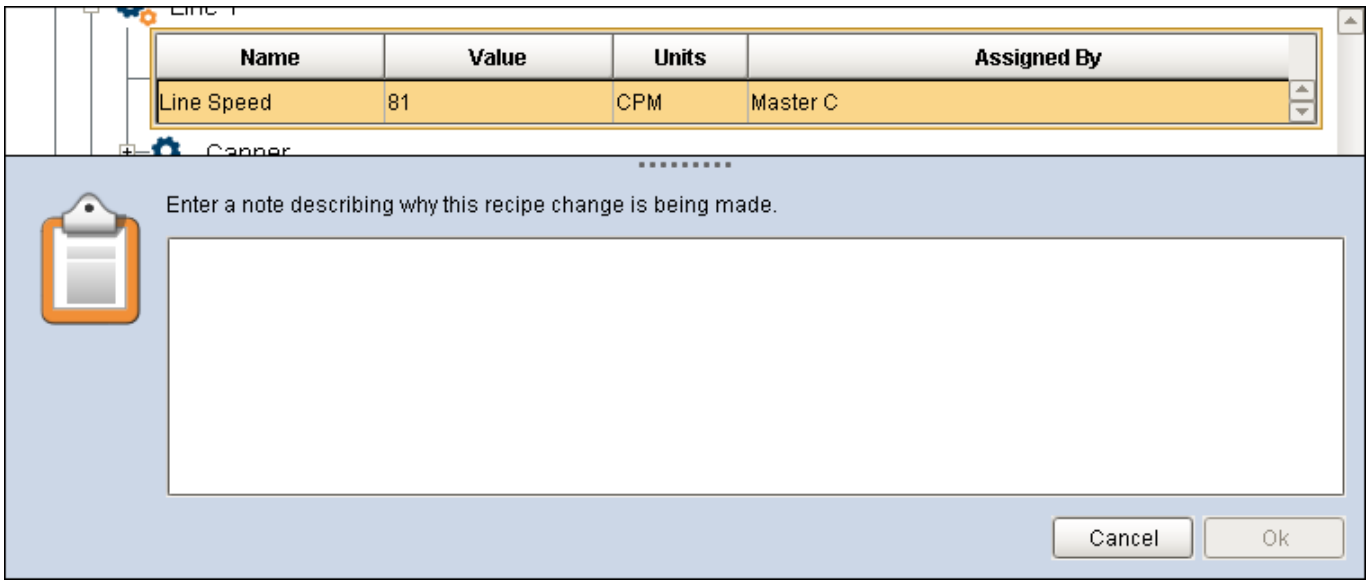
Sub recipes can also be managed by setting both the Show Item Defaults and Show Sub Recipes properties to true. See Sub Recipes for more information. New sub recipes can be added by right clicking on the Sub Recipes node and selecting Add Sub Recipe menu item. Sub recipes can also be removed, renamed or etc. by right clicking on the node of a sub recipe and selecting the desired menu item. The Default sub recipe is always shown and cannot be renamed or deleted. It is reserved for holding the default values for a machine.



© Inductive Automation

Sub Recipe Editor

Based on the setting of the Require Note property, notes are required any time changes are made to a recipe, sub recipe or default values. The note panels is shown below and the appearance is defined by several properties. The Popup Panel Font property determines the font of the text, the Note Panel Icon Path property determines the image on the upper left hand corner and the Note Background Color property determines the background color. This is just an example of the many properties that change the appearance of the Recipe Editor component.



Recipe Editor Note

The other properties that control what in shown and what operations are allowed are described below.

Properties

This component has standard Ignition properties with the addition of the following properties:

Show Recipes

Set to true to show recipes in the editor. This determines if the root Recipes node is shown in the recipe editor.

```
Scripting name          showRecipes
Data Type               Boolean
```

Show Master Recipes

Set to true to show master recipes in the editor. The Show Recipes property must also be set to true.

```
Scripting name          showMasterRecipes
Data Type               Boolean
```

Show Descendants

Set to true to show recipe descendants (or child recipes). It determines if the Descendants tree node is show in each recipe.

```
Scripting name          showDescendants
Data Type               Boolean
```

© Inductive Automation

Show Recipe Items

Set to true to show production items under the recipe node.

```
Scripting name          showRecipeItems
Data Type               Boolean
```

Show Item Children

Set to true to show child production items of the main production items under the recipe node. For example, if Line 1 is added to a recipe and the Show Item Children is set to true, then the cells, cell groups and locations will be shown under Line 1. If the Show Item Children is set to false, then only Line 1 will be shown.

```
Scripting name          showItemChildren
Data Type               Boolean
```

Show Values

Set to true to show the recipe value table in the recipe editor.

```
Scripting name          showValues
Data Type               Boolean
```

Show Item Defaults

Set to true to show the item default values and sub recipes in the recipe editor.

```
Scripting name          showItemDefaults
Data Type               Boolean
```

Show Sub Recipes

Set to true to show sub recipes in the recipe editor. The default values will show up as a sub recipe.

```
Scripting name          showItemDefaults
Data Type               Boolean
```

Enable Recipe Editing

Set to true to enable recipe editing.

```
Scripting name          enableRecipeEditing
Data Type               Boolean
```

Enable Security Editing

Set to true to enable recipe value security editing.

```
Scripting name          enableSecurityEditing
Data Type               Boolean
```

Read Only

Set to true to disable editing of everything. The recipe editor will be in view only mode.

```
Scripting name          readOnly
Data Type               Boolean
```

© Inductive Automation

Require Note

Set to true to require a note for any changes. If the user does not enter a note, then the change they are requesting will be canceled.

```
Scripting name          requireNote
Data Type               Boolean
```

Item Path Filter

To limit which production items to show in the recipe editor, this property can be set. This provides a method to only show production items that are of interest to the end user. The wildcard characters "*" or "?" can also be included in the filter value.

Example
Enterprise\Site\Area 1\Line 1

```
Scripting name          itemPathFilter
Data Type               String
```

Recipe Name Filter

To limit which recipe to show in the recipe editor, this property can be set. This provides a method to only show recipes that are of interest to the end user. The wildcard characters "*" or "?" can also be included in the filter value.

Example
Recipe C*

```
Scripting name          recipeNameFilter
Data Type               String
```

Recipe Value Name Filter

To limit which recipe values for the recipe to show in the recipe editor, this property can be set. This provides a method to only show recipe values that are of interest to the end user. The wildcard characters "*" or "?" can also be included in the filter value.

Example
Recipe C*

```
Scripting name          recipeValueNameFilter
Data Type               String
```

Recipe Value Category

Category of recipe values to return. Where 1 is recipe values created by the recipe module, 2 is recipe values created by the OEE module and 3 is recipe values created by the SPC module. Use blank to include all categories.

```
Scripting name          recipeValueCategory
Data Type               String
```

Default Row Height

Set to the default row height of items in the recipe editor tree.

```
Scripting name          defaultRowHeight
Data Type               Integer
```

© Inductive Automation

Max Recipe Value Rows

Set to the maximum number of rows to display in the recipe value table before scrolling.

```
Scripting name          maxRecipeValueRows
Data Type               Integer
```

Popup Panel Font

Set to the font to show text on the note, security and production item selector popup panels.

```
Scripting name          popupPanelFont
Data Type               Font
```

Value Table Font

Set to the font to show text in the recipe value table.

```
Scripting name          valueTableFont
Data Type               Font
```

Value Table Header Font

Set to the font to show text in the recipe value table header.

```
Scripting name          valueTableHeaderFont
Data Type               Font
```

Recipes Icon Path

Set to the Ignition image path of the icon to use for the root Recipes node. If this property is left blank, the default icon will be used.

```
Scripting name          recipesIconPath
Data Type               String
```

Recipe Icon Path

Set to the Ignition image path of the icon to use for recipe nodes. If this property is left blank, the default icon will be used.

```
Scripting name          recipeIconPath
Data Type               String
```

Recipe Descendants Icon Path

Set to the Ignition image path of the icon to use for recipe descendant nodes. If this property is left blank, the default icon will be used.

```
Scripting name          recipeDescendantsIconPath
Data Type               String
```

Default Values Icon Path

Set to the Ignition image path of the icon to use for default values nodes. If this property is left blank, the default icon will be used.

```
Scripting name          defaultValuesIconPath
Data Type               String
```

Sub Recipes Icon Path

Set to the Ignition image path of the icon to use for root sub recipe nodes. If this property is left blank, the default icon will be used.

```
Scripting name          subRecipesIconPath
Data Type               String
```

© Inductive Automation

Sub Recipe Icon Path

Set to the Ignition image path of the icon to use for sub recipe nodes. If this property is left blank, the default icon will be used.

```
Scripting name          subRecipeIconPath
Data Type               String
```

Default SubRecipe Icon Path

Set to the Ignition image path of the icon to use for default sub recipe nodes. If this property is left blank, the default icon will be used.

```
Scripting name          defaultSubRecipeIconPath
Data Type               String
```

Prod Item Icon Path

Set to the Ignition image path of the icon to use for production item nodes. If this property is left blank, the default icon will be used.

```
Scripting name          prodItemIconPath
Data Type               String
```

Menu Add Icon Path

Set to the Ignition image path of the icon to use for add menu items. If this property is left blank, the default icon will be used.

```
Scripting name          menuAddIconPath
Data Type               String
```

Menu Rename Icon Path

Set to the Ignition image path of the icon to use for rename menu items. If this property is left blank, the default icon will be used.

```
Scripting name          menuRenameIconPath
Data Type               String
```

Menu Delete Icon Path

Set to the Ignition image path of the icon to use for delete menu items. If this property is left blank, the default icon will be used.

```
Scripting name          menuDeleteIconPath
Data Type               String
```

Menu Revert Icon Path

Set to the Ignition image path of the icon to use for revert value menu items. If this property is left blank, the default icon will be used.

```
Scripting name          menuRevertIconPath
Data Type               String
```

Menu Security Icon Path

Set to the Ignition image path of the icon to use for security menu items. If this property is left blank, the default icon will be used.

```
Scripting name          menuSecurityIconPath
Data Type               String
```

© Inductive Automation

Menu Select Items Icon Path

Set to the Ignition image path of the icon to use for select production items menu items. If this property is left blank, the default icon will be used.

Scripting name	menuSelectItemsIconPath
Data Type	String

Note Panel Icon Path

Set to the Ignition image path of the icon to use display on the note slide out panel. If this property is left blank, the default icon will be used.

Scripting name	notePanelIconPath
Data Type	String

Security Panel Icon Path

Set to the Ignition image path of the icon to use display on the recipe value security slide out panel. If this property is left blank, the default icon will be used.

Scripting name	securityPanelIconPath
Data Type	String

Item Select Panel Icon Path

Set to the Ignition image path of the icon to display on the select production item slide out panel. If this property is left blank, the default icon will be used.

Scripting name	itemSelectPanelIconPath
Data Type	String

Note Background Color

Set the background color to use on the note slide out panel.

Scripting name	noteBackgroundColor
Data Type	Color

Security Background Color

Set the background color to use on the security slide out panel.

Scripting name	securityBackgroundColor
Data Type	Color

Item Selector Background Color

Set the background color to use on the select production item slide out panel.

Scripting name	itemSelectorBackgroundColor
Data Type	Color

© Inductive Automation

User Menu Items

A dataset containing user menu item to show in popup menus within the recipe editor component.

```
Scripting name              userMenuItems
Data Type                   Dataset
```

The Dataset must have the following columns that are reference by column number when building the user menus:

0	String	Node type that determines which popup menu the user menu item will appear in. Valid options are: Root Recipe Recipe Descendants Production Item Root Sub Recipe Sub Recipe
1	String	Text to appear for the user menu item. This is also used to identify the user menu item that was clicked in the userMenuItemClicked event.
2	String	Ignition image path of the icon to display in the user menu item.

Events

meunu - userMenuItemClicked

Is fired whenever a user menu item is selected.

Event Properties

event.getMenuItemName()
Returns the name of the user menu item that triggered the event.
Data Type String

event.getSelectedItemPath()
Returns the item path of the currently selected production item.
Data Type String

event.getSelectedRecipe()
Returns the name of the currently selected recipe.
Data Type String

event.getSelectedValueName()
Returns the name of the currently selected recipe value.
Data Type String

event.getSelectedSubRecipe()
Returns the name of the currently selected sub recipe.
Data Type String

event.getUser()
Returns the name of the logged in user.
Data Type String

event.isShowRecipes()
Returns true if Show Recipes property is true.
Data Type Boolean

event.isShowItemDefaults()
Returns true if Show Item Defaults property is true.
Data Type Boolean

© Inductive Automation

Methods

changeLocalizationString(key, displayText)

Any of the text that is displayed in the recipe editor change be changed. For example, displaying Recipes for the root recipe node can be replaced with Products. This can be done for any static text in the recipe editor including menu items.

parameters

`key` — The key to the string value to change.
Data Type String

Recipe Editor component keys:

```
node.recipes
node.default.values
node.subrecipes
node.subrecipes.default
node.descendants
node.subrecipes.default

menu.recipe.add
menu.recipe.delete
menu.recipe.rename
menu.value.revert
menu.recipe.revertvalues
menu.value.read
menu.recipe.setvalues
menu.value.security
menu.recipe.selectitems
menu.subrecipe.add
menu.subrecipe.delete
menu.subrecipe.rename
menu.recipe.import
menu.recipe.export

panel.item.select.inst
panel.security.inst
panel.note.inst
panel.cancel.desc
panel.ok.desc
```

`displayText` — The new text to replace the existing display text.
Data Type String

returns nothing

Sample script to change the root Recipes node text to Spanish.

Script from internalFrameActivated event on the window

```
system.gui.getParentWindow(event).getComponentForPath('Root Container.Recipe TreeView').changeLocalizatio
```

4.4.2 Recipe Changelog Viewer

Description

A component that is added to Ignition windows to display recipe change log history in a table. This is just one method of viewing a recipe change log history and for more information on the other methods see the Recipe Change Log section.

The Recipe Change Log Viewer component is inherited from the Ignition Table component and many of the features are carried through to this component. This component simplifies displaying a recipe changes log by handling all of the backend database queries based on the property settings of the component. The appearance and columns to display are changed using the table customizer that is accessed by right clicking on the Recipe Change Log Viewer component and selecting Customizers->Table Customizer.

Properties

This component has standard Ignition properties with the addition of the following properties:

Item Path Filter

To limit which production items to show in the recipe change log history, this property can be set. This provides a method to only show production items that are of interest to the end user. The wildcard characters "*" or "?" can also be included in the filter value.

Example
Enterprise\Site\Area 1\Line 1

```
Scripting name          itemPathFilter
Data Type               String
```

Recipe Name Filter

To limit which recipes to show in the recipe change log history, this property can be set. This provides a method to only show recipes that are of interest to the end user. The wildcard characters "*" or "?" can also be included in the filter value.

Example
Recipe C*

```
Scripting name          recipeNameFilter
Data Type               String
```

Value Name Filter

To limit which recipe values to show in the recipe change log history, this property can be set. This provides a method to only show recipe values that are of interest to the end user. The wildcard characters "*" or "?" can also be included in the filter value.

```
Scripting name          valueNameFilter
Data Type               String
```

© Inductive Automation

User Filter

To limit which users to show in the recipe change log history, this property can be set. This provides a method to only show changes made by a user. The wildcard characters "*" or "?" can also be included in the filter value.

```
Scripting name          userFilter
Data Type               String
```

Sub Product Code Filter

To limit which sub product codes to show in the recipe change log history, this property can be set. This provides a method to only show sub product codes that are of interest to the end user. The wildcard characters "*" or "?" can also be included in the filter value.

```
Scripting name          subProductCodeFilter
Data Type               String
```

Show Recipe Changes

If true show recipe changes.

```
Scripting name          showRecipeChanges
Data Type               Boolean
```

Show Default Value Changes

If true show production item default value and sub recipe changes.

```
Scripting name          showdefaultValueChanges
Data Type               Boolean
```

Start Date

Starting date of any entries in the recipe change log to include.

```
Scripting name          startDate
Data Type               Date
```

End Date

Ending date of any entries in the recipe change log to include.

```
Scripting name          endDate
Data Type               Date
```

Date

Change log historical data that can be bound to or used in script.

```
Scripting name          date
Data Type               Dataset
```

Events

none

Methods

none

4.4.3 Recipe Variance Viewer

Description

A component that is added to Ignition windows to display recipe variances in a table. This is just one method of viewing a recipe variances and for more information on the other methods see the Variance Monitoring section. The Recipe Variance Viewer is automatically updated when live recipe value variances are detected by the Recipe / Changeover Module. The Show Full Details property will cause the initial values when the recipe was selected, to also be included.

The Recipe Variance Viewer component is inherited from the Ignition Table component and many of the features are carried through to this component. This component simplifies displaying recipe variances by handling all of the backend database queries based on the property settings of the component. The appearance and columns to display are changed using the table customizer that is accessed by right clicking on the Recipe Variance Viewer component and selecting Customizers->Table Customizer.

Properties

This component has standard Ignition properties with the addition of the following properties:

Item Path Filter

This is a required property to limit the variances to a single production item.

Example
Enterprise\Site\Area 1\Line 1

```
Scripting name          itemPathFilter
Data Type               String
```

Recipe Name Filter

To limit which recipes to show in the variances, this property can be set. This provides a method to only show recipes that are of interest to the end user. The wildcard characters "*" or "?" can also be included in the filter value.

Example
Recipe C*

```
Scripting name          recipeNameFilter
Data Type               String
```

Value Name Filter

To limit which recipe values to show in the variances, this property can be set. This provides a method to only show recipe values that are of interest to the end user. The wildcard characters "*" or "?" can also be included in the filter value.

```
Scripting name          valueNameFilter
Data Type               String
```

© Inductive Automation

Sub Recipe Name Filter

To limit which sub recipes to show in the variances, this property can be set. This provides a method to only show sub product codes that are of interest to the end user. The wildcard characters "*" or "?" can also be included in the filter value.

```
Scripting name          subRecipeNameFilter
Data Type               String
```

Show Full Details

If true, show recipe values when the recipe was first selected.

```
Scripting name          showFullDetails
Data Type               Boolean
```

Display Variance Type

Set the variance types to include in the results. Valid values are:

- ❑ RECIPE to return variances that occurred while a production item was selected to a recipe.
- ❑ SUB_RECIPE to return variances that occurred while a sub product code was selected for a production item. See Sub Recipes for more information.

```
Scripting name          showdefaultValueChanges
Data Type               VarianceEntryTypes
```

Display Variance Scope

Set the variance scope to include in the results. Valid values are:

- ❑ LAST to return variances that occurred for the current or last recipe that a production item was set. This is useful for detecting any variances in real time for a production run. If the production run has stopped, it will return the variances as long as a new recipe has not been selected for the production item.
- ❑ DATE_RANGE to return variances for the date range specified with the Start Date and End Date properties.

```
Scripting name          displayVarianceScope
Data Type               VarianceScopeType
```

Start Date

Starting date of any entries in the recipe change log to include. The Display Variance Scope property must be set to DATE_RANGE.

```
Scripting name          startDate
Data Type               Date
```

End Date

Ending date of any entries in the recipe change log to include. The Display Variance Scope property must be set to DATE_RANGE.

```
Scripting name          endDate
Data Type               Date
```

Data

Recipe value variances data that can be bound to or used in script.

```
Scripting name          date
Data Type               Dataset
```

© Inductive Automation

Events
none

Methods
none

4.4.4 Recipe Selector Combo

Description
A component that is added to Ignition windows to select recipes in a drop down list. This is just one method of selecting a recipe for a production line, cell, cell group or location. For more information on the other methods see the Selecting Recipes section. The Recipe Selector Combo component is automatically updated when recipes are added, removed, etc.

Properties
This component has standard Ignition properties with the addition of the following properties:

Item Path

This is a required property to specify a single production item to show recipes for.

Example
Enterprise\Site\Area 1\Line 1

```
Scripting name          itemPath
Data Type               String
```

Recipe Name Filter

To limit which recipes to show, this property can be set. This provides a method to only show recipes that are of interest to the end user. The wildcard characters "*" or "?" can also be included in the filter value.

Example
Recipe C*

```
Scripting name          recipeNameFilter
Data Type               String
```

Selected Recipe Name

This will reflect the recipe that the user selected. It can also be set using script.

```
Scripting name          selectedRecipeName
Data Type               String
```

Events
none

Methods
none

© Inductive Automation

4.4.5 Recipe Selector List

Description

A component that is added to Ignition windows to select recipes in a scrollable list. This is just one method of selecting a recipe for a production line, cell, cell group or location. For more information on the other methods see the Selecting Recipes section. The Recipe Selector List component is automatically updated when recipes are added, removed, etc. The current recipe selected is also automatically updated if the recipe changes from a different source.

To select a recipe for a production item, the user can right click on a recipe then select the Select Recipe menu item. To cancel a recipe, right click on the selected recipe then select the Cancel Recipe menu item. This component also supports additional menu item to be added by using the User Menu Items property and the userMenuItemClicked event.

Properties

This component has standard Ignition properties with the addition of the following properties:

Item Path

This is a required property to specify a single production item to show recipes for.

Example
Enterprise\Site\Area 1\Line 1

```
Scripting name          itemPath
Data Type               String
```

Recipe Name Filter

To limit which recipes to show, this property can be set. This provides a method to only show recipes that are of interest to the end user. The wildcard characters "*" or "?" can also be included in the filter value.

Example
Recipe C*

```
Scripting name          recipeNameFilter
Data Type               String
```

Selected Recipe Name

This will reflect the recipe that the user selected. It can also be set using script.

```
Scripting name          selectedRecipeName
Data Type               String
```

Selected Icon Path

Set to the Ignition image path of the icon to use to indicate the selected recipe. If this property is left blank, the default icon will be used

```
Scripting name          selectedIconPath
Data Type               String
```

© Inductive Automation

User Menu Items

A dataset containing user menu item(s) to show in popup menus within the recipe selector list component.

```
Scripting name          userMenuItems
Data Type               Dataset
```

The Dataset must have the following columns that are reference by column number when building the user menus:

0	String	Text to appear for the user menu item. This is also used to identify the user menu item that was clicked in the userMenuItemClicked event.
1	String	Ignition image path of the icon to display in the user menu item.

Events

menu - userMenuItemClicked Is fired whenever a user menu item is selected.

Event Properties

event.getMenuItemName() Returns the name of the user menu item that triggered the event.
Data Type String

event.getSelectedItemPath() Returns the item path of the production item.
Data Type String

event.getSelectedRecipe() Returns the name of the currently selected recipe.
Data Type String

Methods

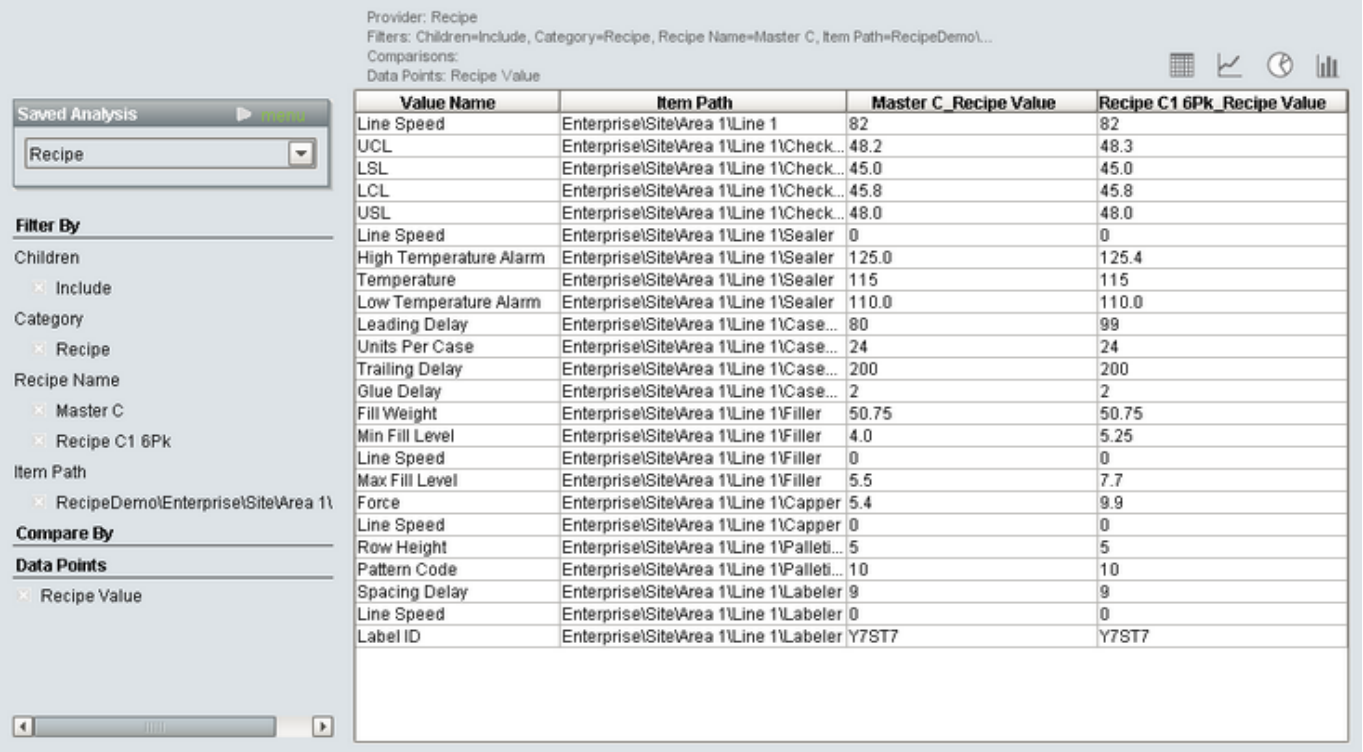
none

4.5 Analysis Providers

Each of the MES modules depend on the core Production Module. When any of the MES modules are purchased, the Production Module is included at no additional cost. The Production Module provides the production model functionality that is an object oriented hierarchy of production facilities. It also provides analysis functionality that each specific MES module can extend. The Recipe / Changeover Module provides four analysis providers to analyze differences between recipes, variances and change logs.

The image below shows the impromptu analysis screen where a Recipe C1 6Pk is being compared to its parent Master C. Filters and data points can be selected using the Production Analysis Selector component, but the Analysis Controller will allow requesting comparisons of recipes without the user interface. The filters and data points are defined through the component properties and is how data for reports is collected.

© Inductive Automation



Value Name	Item Path	Master C_Recipe Value	Recipe C1 6Pk_Recipe Value
Line Speed	Enterprise\Site\Area 1\Line 1	82	82
UCL	Enterprise\Site\Area 1\Line 1\Check...	48.2	48.3
LSL	Enterprise\Site\Area 1\Line 1\Check...	45.0	45.0
LCL	Enterprise\Site\Area 1\Line 1\Check...	45.8	45.8
USL	Enterprise\Site\Area 1\Line 1\Check...	48.0	48.0
Line Speed	Enterprise\Site\Area 1\Line 1\Sealer	0	0
High Temperature Alarm	Enterprise\Site\Area 1\Line 1\Sealer	125.0	125.4
Temperature	Enterprise\Site\Area 1\Line 1\Sealer	115	115
Low Temperature Alarm	Enterprise\Site\Area 1\Line 1\Sealer	110.0	110.0
Leading Delay	Enterprise\Site\Area 1\Line 1\Case...	80	99
Units Per Case	Enterprise\Site\Area 1\Line 1\Case...	24	24
Trailing Delay	Enterprise\Site\Area 1\Line 1\Case...	200	200
Glue Delay	Enterprise\Site\Area 1\Line 1\Case...	2	2
Fill Weight	Enterprise\Site\Area 1\Line 1\Filler	50.75	50.75
Min Fill Level	Enterprise\Site\Area 1\Line 1\Filler	4.0	5.25
Line Speed	Enterprise\Site\Area 1\Line 1\Filler	0	0
Max Fill Level	Enterprise\Site\Area 1\Line 1\Filler	5.5	7.7
Force	Enterprise\Site\Area 1\Line 1\Capper	5.4	9.9
Line Speed	Enterprise\Site\Area 1\Line 1\Capper	0	0
Row Height	Enterprise\Site\Area 1\Line 1\Palleti...	5	5
Pattern Code	Enterprise\Site\Area 1\Line 1\Palleti...	10	10
Spacing Delay	Enterprise\Site\Area 1\Line 1\Labeler	9	9
Line Speed	Enterprise\Site\Area 1\Line 1\Labeler	0	0
Label ID	Enterprise\Site\Area 1\Line 1\Labeler	Y7ST7	Y7ST7

Recipe Analysis Provider

Below describes each of the analysis providers included in the Recipe / Changeover Module.

4.5.1 Recipe Analysis Provider

The recipe analysis provider is used to collect recipe values for display or report purposes. By adding multiple Recipe Name filter values, the values will be returned for each recipe name allowing comparing of two or more recipes.

Filters

The recipe analysis provider can accept the following filters:

Category

This is a required filter to specify the type of recipes to return. Only one of the valid options are required:

- ❑ Recipe
- ❑ Sub Recipe - This includes default values or sub recipes for production items.

Example when using it with the Analysis Controller:
Category=Recipe

Item Path

This is a required filter to specify the production item to include in the results. It is the item path for the desired item path(s). Because analysis is independent of projects, the project name is required in the item path.

Example when using it with the Analysis Controller:
Item Path=RecipeDemo\Enterprise\Site\Area 1\Line 1, Item Path=RecipeDemo\Enterprise\Site\Area 1\Line 2

Children

This is a filter to specify if children of the production item(s) specified in the Item Path filter should be included.
Only one of the valid options are required:
- ❑ Include
- ❑ Exclude (Default)

Example when using it with the Analysis Controller:
Children=Include

Format

This is a filter to specify the format of the results.
Only one of the valid options are required:
- ❑ None - A row in the results will be created for each recipe included in the Recipe Name filter.
- ❑ Recipe Comparison (Default) - This format creates a recipe value column for each recipe included in the Recipe Name filter and consolidates the item path, value name, etc. columns that are common.

Example when using it with the Analysis Controller:
Format=None

Column Naming

This is a filter to specify how to name the recipe value columns when the Format filter is set to Recipe Comparison (Default). When this filter is set to Recipe Name Prefix (Default), it is difficult to create a recipe comparison report because the column names change depending on the recipes being compared. Setting this filter to Number Suffix will cause the recipe value column names to always be the same, which simplifies reports.
Only one of the valid options are required:
- ❑ Number Suffix
- ❑ Recipe Name Prefix (Default)

Example when using it with the Analysis Controller:
Column Naming=Number Suffix

Recipe Name

This is a filter to limit the recipe(s) to include in the results.

Example when using it with the Analysis Controller:
Recipe Name=Recipe A, Recipe Name=Recipe B

© Inductive Automation

Recipe Value Name
This is a filter to limit the recipe value(s) to include in the results.

Example when using it with the Analysis Controller:
Recipe Value Name=Line Speed, Recipe Value Name=Force

Value Types
This is a filter to specify the type of recipe values to include.
Only one of the valid options are required:
- ❑ Equal Values - Include only recipe values that match between two or more recipes.
- ❑ Not Equal Values - Include only recipe values that do not match between two or more recipes.
- ❑ All Values (Default) - Include both values that do not match and do match between two or more recipes.

Example when using it with the Analysis Controller:
Value Types=Not Equal Values

Compare By
The recipe analysis provider does not allow any comparison statements.

Data Points
The recipe analysis provider can accept the following data points:

Assigned By
This is the recipe or production item that assigned the recipe value based on inheritance.

Data Type
This is the data type of the recipe value.

Description
This is the description from the recipe value configuration that was entered in the designer.

Format
This is the numeric format from the associated Ignition tag.

Recipe Name
This is the name of the recipe. If the Format filter is set to Recipe Comparison (Default), then this data point will not be included in the results. This is because columns are added for each recipe being compared. For example when comparing Recipe A to Recipe B, there will be Recipe_A_Recipe_Value and Recipe_B_Recipe_Value columns.

Units
This is the units from the associated Ignition tag.

4.5.2 Recipe Variance Analysis Provider

The recipe variance analysis provider is used to collect recipe variances for display or report purposes. This provider can be used to collect variances in real-time or historically for a date range.

Filters

The recipe variance analysis provider can accept the following filters:

Scope

This is a required filter to specify the scope of what to include in the results:
- ❑ Date Range
- ❑ Last Active Recipe - This will look for the last recipe selection in the variance log and only include the associated variances in the results. This can be used to monitor active runs or the last run if the recipe has been canceled.

Example when using it with the Analysis Controller:
Scope=Last Active Recipe

Item Path

This is a required filter to specify the production item to include in the results. It is the item path for the desired item path(s). Because analysis is independent of projects, the project name is required in the item path.

Example when using it with the Analysis Controller:
Item Path=RecipeDemo\Enterprise\Site\Area 1\Line 1, Item Path=RecipeDemo\Enterprise\Site\Area 1\Line 2

Children

This is a filter to specify if children of the production item(s) specified in the Item Path filter should be included.
Only one of the valid options are required:
- ❑ Include
- ❑ Exclude (Default)

Example when using it with the Analysis Controller:
Children=Include

Recipe Name

This is a filter to limit the recipe(s) to include in the results.

Example when using it with the Analysis Controller:
Recipe Name=Recipe A, Recipe Name=Recipe B

Recipe Value Name

This is a filter to limit the recipe value(s) to include in the results.

Example when using it with the Analysis Controller:
Recipe Value Name=Line Speed, Recipe Value Name=Force

Values

This is a filter to specify the type of recipe values to include.
Only one of the valid options are required:
- ❑ Initial Values - Include only the initial recipe values when the recipe was first selected.
- ❑ Changed Values - Include only recipe values that changed after the initial values were set.
- ❑ Both - Include both initial and changed values.

Example when using it with the Analysis Controller:
Values=Both

© Inductive Automation

Compare By

The recipe variance analysis provider does not allow any comparison statements.

Data Points

The recipe variance analysis provider can accept the following data points:

Description	This is the description from the recipe value configuration that was entered in the designer.
From Value	This is the value of the Ignition tag associated with the recipe value before it changed
Item Path	This is the item path of the production item for the recipe value.
Recipe Name	This is the name of the recipe at the time when the recipe value changed.
Recipe Value	The value that is defined in the recipe.
Time Stamp	The date and time the recipe value changed.
To Value	This is the value of the Ignition tag associated with the recipe value after it changed.
Units	This is the units from the associated Ignition tag.
Value Name	This is the name from the recipe value configuration that was entered in the designer.

4.5.3 Sub Product Code Variance Analysis Provider

The sub product code variance analysis provider is used to collect sub product code variances for display or report purposes. This provider can be used to collect variances in real-time or historically for a date range.

Filters

The recipe variance analysis provider can accept the following filters:

Scope

This is a required filter to specify the scope of what to include in the results:
- ❑ Date Range
- ❑ Last Active Sub Recipe - This will look for the last sub product code selection in the variance log and only include the associated variances in the results. This can be used to monitor active runs or the last run if the sub recipe has been canceled.

Example when using it with the Analysis Controller:
Scope=Last Active Sub Recipe

Item Path

This is a required filter to specify the production item to include in the results. It is the item path for the desired item path(s). Because analysis is independent of projects, the project name is required in the item path.

Example when using it with the Analysis Controller:
Item Path=RecipeDemo\Enterprise\Site\Area 1\Line 1, Item Path=RecipeDemo\Enterprise\Site\Area 1\Line 2

Children

This is a filter to specify if children of the production item(s) specified in the Item Path filter should be included.
Only one of the valid options are required:
- ❑ Include
- ❑ Exclude (Default)

Example when using it with the Analysis Controller:
Children=Include

Recipe Value Name

This is a filter to limit the recipe value(s) to include in the results.

Example when using it with the Analysis Controller:
Recipe Value Name=Line Speed, Recipe Value Name=Force

Sub Recipe Name

This is a filter to limit the sub recipe(s) to include in the results.

Example when using it with the Analysis Controller:
Recipe Name=1A, Recipe Name=1B

Values

This is a filter to specify the type of recipe values to include.
Only one of the valid options are required:
- ❑ Initial Values - Include only the initial recipe values when the recipe was first selected.
- ❑ Changed Values - Include only recipe values that changed after the initial values were set.
- ❑ Both - Include both initial and changed values.

Example when using it with the Analysis Controller:
Values=Both

© Inductive Automation

Compare By

The sub product code variance analysis provider does not allow any comparison statements.

Data Points

The sub product code variance analysis provider can accept the following data points:

Description	This is the description from the recipe value configuration that was entered in the designer.
From Value	This is the value of the Ignition tag associated with the recipe value before it changed
Item Path	This is the item path of the production item for the recipe value.
Sub Recipe Name	This is the name of the sub recipe at the time when the recipe value changed.
Recipe Value	The value that is defined in the recipe.
Recipe Value Name	This is the name from the recipe value configuration that was entered in the designer.
Time Stamp	The date and time the recipe value changed.
To Value	This is the value of the Ignition tag associated with the recipe value after it changed.
Units	This is the units from the associated Ignition tag.

4.5.4 Recipe Change Log Analysis Provider

The recipe change log analysis provider is used to collect the change log entries for display or report purposes. All details of changes made to recipes including adding new recipes, adding production items to recipes and much more can be returned using the recipe change log analysis provider.

Filters

The recipe change log analysis provider can accept the following filters:

© Inductive Automation

Category

This is a required filter to specify the type of recipes to return. One or more of the valid options are required:
- ❑ Recipe - This includes all recipe changes excluding value changes.
- ❑ Recipe Value - This includes only recipe value changes.
- ❑ Sub Recipe - This includes default value or sub recipe changes excluding value changes.
- ❑ Sub Recipe Value - This includes default value or sub recipe value changes.

Example when using it with the Analysis Controller:
Category=Recipe, Category=Recipe Value

Item Path

This is a required filter to specify the production item to include in the results. It is the item path for the desired item path(s). Because analysis is independent of projects, the project name is required in the item path.

Example when using it with the Analysis Controller:
Item Path=RecipeDemo\Enterprise\Site\Area 1\Line 1, Item Path=RecipeDemo\Enterprise\Site\Area 1\Line 2

Children

This is a filter to specify if children of the production item(s) specified in the Item Path filter should be included. Only one of the valid options are required:
- ❑ Include
- ❑ Exclude (Default)

Example when using it with the Analysis Controller:
Children=Include

Recipe Name

This is a filter to limit the recipe(s) to include in the results.

Example when using it with the Analysis Controller:
Recipe Name=Recipe A, Recipe Name=Recipe B

Recipe Value Name

This is a filter to limit the recipe value(s) to include in the results.

Example when using it with the Analysis Controller:
Recipe Value Name=Line Speed, Recipe Value Name=Force

Sub Recipe Name

This is a filter to limit the sub recipe(s) to include in the results.

Example when using it with the Analysis Controller:
Recipe Name=1A, Recipe Name=1B

Compare By

The recipe analysis provider does not allow any comparison statements.

Data Points

© Inductive Automation

The recipe analysis provider can accept the following data points:

Change Type	This is a description of the type of change. For example: it can be Recipe value changed or Recipe value reverted.
Changed By	This is the person that made the change.
Description	This is the description from the recipe value configuration that was entered in the designer.
From Value	This is the value before the change.
Info	This is additional information that further describes the change.
Item Path	This is the item path of the production item that the change was made for.
Note	This is the note that was entered by the user at the time of the change.
Recipe Name	This is the name of the recipe the change was made for.
Sub Product Code	This is the sub product code the change was made for.
Time Stamp	The date and time of the changed.
To Value	This is the value after the change.
Units	This is the units from the associated Ignition tag.
Value Name	This is the name from the recipe value configuration that was entered in the designer.

4.6 Production OPC Values

This references details the production OPC values that the Recipe / Changeover Module provides. For each property, the Ignition data type is listed and if it is read only. The Ignition data types correspond to the data types that are available for SQLTags.

Within this reference, the "Read Only" means that the OPC value cannot be written to through the OPC Production Server. It can only be set in the designer or it is a calculated value. Trying to write to a read

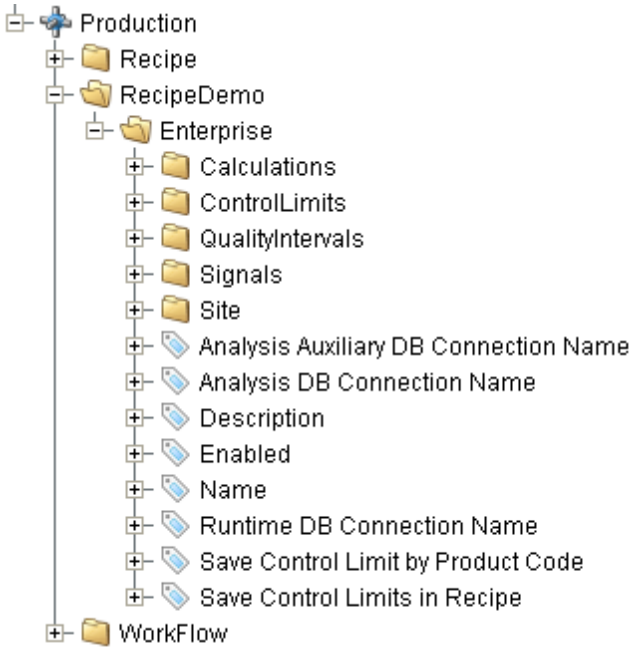
only property will result in an error message being shown.

Depending on the MES modules that are installed into the Ignition server, more or less production OPC values will appear when browsing. For example, if only the Recipe / Changeover Module is installed, then only production OPC values that the core MES or Recipe / Changeover Module provide will appear.

4.6.1 Enterprise

Description

The *enterprise* folder contains some properties associated with the *enterprise* and a folder for each production *Site* within it. The name is the same as the *enterprise* name that is configured in the designer. The image below represents the "Enterprise" of the RecipeDemo project.



Enterprise

Child Folders

Site One folder will exist for each *Site* that has been configured in the Ignition Designer. The folder can be opened to view all values within the *site*.

Properties

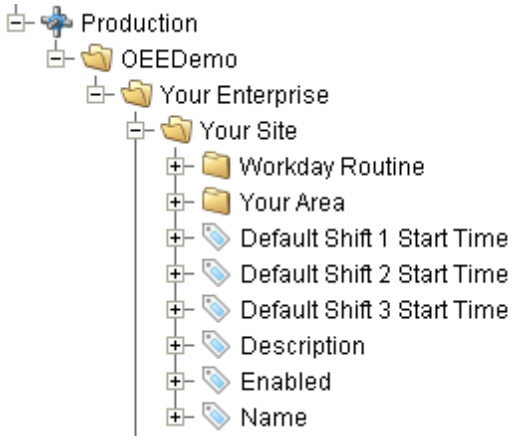
Analysis Auxiliary DB Connection Name	The name of the auxiliary (mirror) analysis database connection. Can be blank if no auxiliary DB connection is configured.	String Read Only
Analysis DB Connection Name	The name of the analysis database connection.	String Read Only
Description	Optionally, this property can be set to a description for the *enterprise*. It is not used by the MES modules other than for reference.	String
Enabled	This reflects the *enterprise* Enabled property in the Designer. If the *enterprise* Enabled is set to true, then the MES production model will perform calculations for the *enterprise* and all *sites*, *areas*, *lines, cells, cell groups and location* within it. If this property is set to false, then none of the *sites*, *areas*, *lines, cells, cell groups and locations* will have calculations performed.	Boolean
Name	This reflects the name of the *enterprise* that is set in the designer.	String Read Only
Runtime DB Connection Name	The name of the runtime database connection.	String Read Only

4.6.2 Site

Description

The *site* folder contains some properties associated with the production *site* and a folder for each production *area* within it. The name is the same as the *site* name that is configured in the designer. The image below represents the "Your Site" of the OEEDemo project.

© Inductive Automation



Site

Child Folders

Area One folder will exist for each area that has been configured in the Ignition Designer. The folder can be opened to view all values within the *area*.

RecipeValue Any recipe values that are configured for the production site will appear in this folder.

Properties

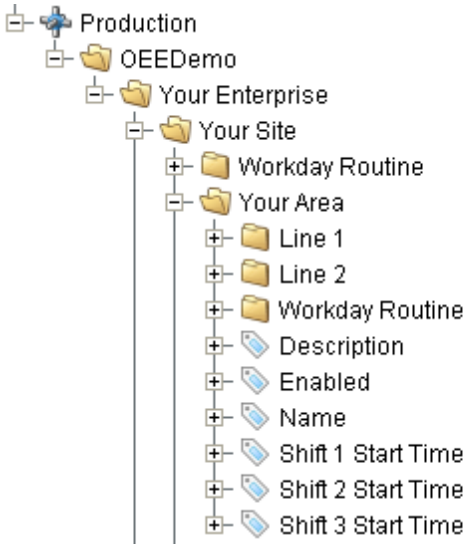
Description	Optionally, this property can be set to a description for the *site*. It is not used by the OEE Downtime and Scheduling Module other than for reference.	String
Enabled	This reflects the *site* Enabled property in the Designer. If the *site* Enabled is set to true, then the OEE Downtime and Scheduling module will perform calculations for the *site* and all *areas*, *lines* and *cells* within it. If this property is set to false, then none of the *areas*, *lines* or *cells* will have calculations performed.	Boolean
Name	This reflects the name of the *site* that is set in the designer.	String Read Only

4.6.3 Area

Description

The *area* folder contains some properties associated with the production *area* and a folder for each production *line* within it. The name is the same as the *area* name that is configured in the designer. The image below represents the "Your Area" of the OEEDemo project.

© Inductive Automation



Area

Child Folders

Line	One folder will exist for each *Line* that has been configured in the Ignition Designer. The folder can be opened to view all values within the *line*.
RecipeValue	Any recipe values that are configured for the production area will appear in this folder.

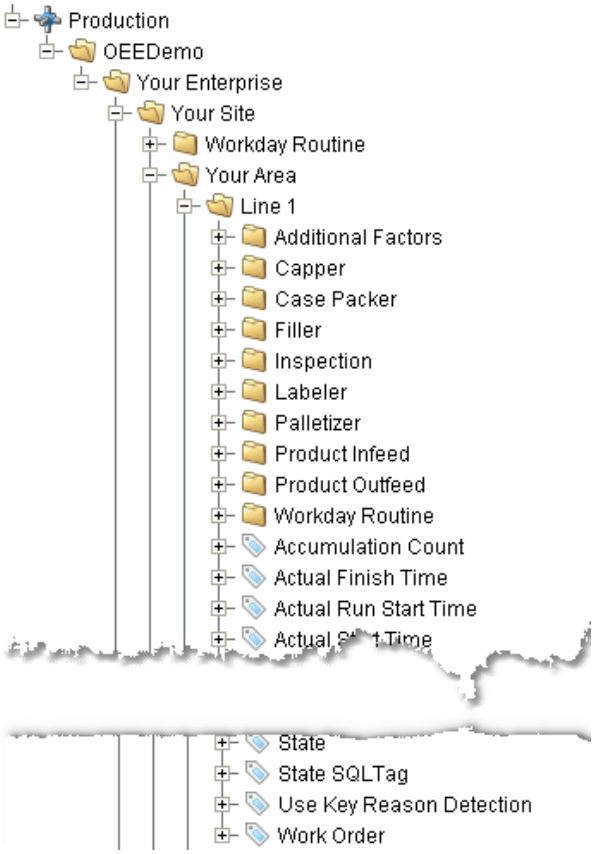
Properties

Description	Optionally, this property can be set to a description for the *area*. It is not used by the OEE Downtime and Scheduling Module other than for reference.	String
Enabled	This reflects the *site* Enabled property in the Designer. If the *area* Enabled is set to true, then the OEE Downtime and Scheduling module will perform calculations for the *area* and all *lines* and cell within it. If this property is set to false, then none of the *lines* or cells will have calculations performed.	Boolean
Name	This reflects the name of the *area* that is set in the designer.	String Read Only

4.6.4 Line

Description

The *line* folder contains some properties associated with the production *line* and a folder for each production cell within it. The name is the same as the *line* name that is configured in the designer. The image below represents the "Line 1" of the OEEDemo project.



Line

Child Folders

RecipeValue Any recipe values that are configured for the production line will appear in this folder.

Cell One folder will exist for each ***Cell*** that has been configured in the Ignition Designer. The folder can be opened to view all values within the cell.

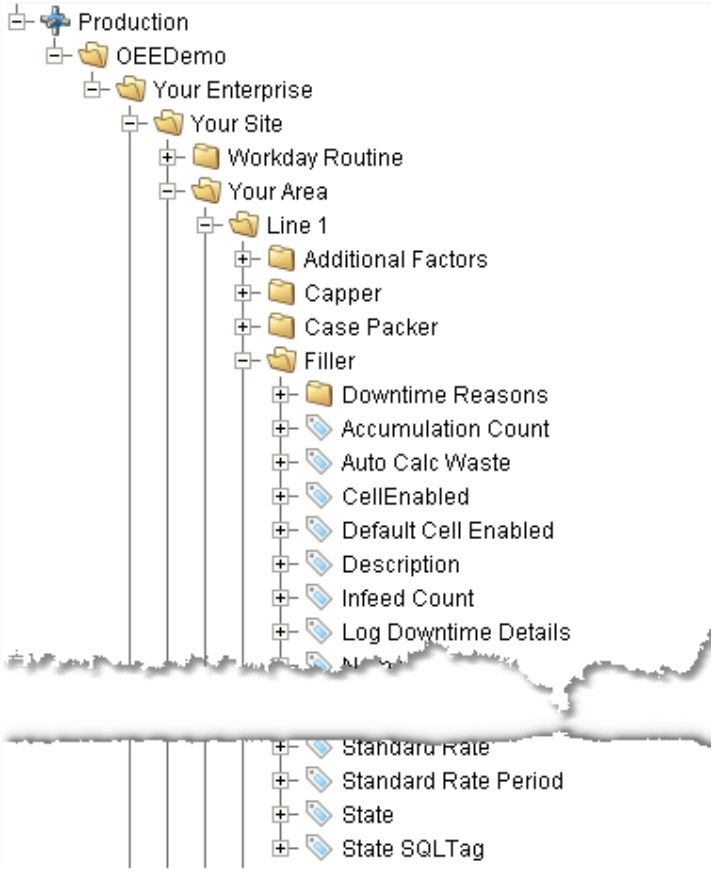
Properties

ActiveRecipeName	If a recipe is active for this production line, then this is the name of the recipe. If a recipe is not active, then this is blank.	String Read Only
RecipeLoading	True if a recipe is currently being loaded for the production line.	Boolean Read Only
RecipeActive	Indicates if a recipe is currently active.	Boolean Read Only
RecipeScale	Set this to the amount to scale a recipe prior to selecting a recipe for the production line.	Double
RecipeTrackingUUID	This is a unique value used for tracking initial recipe values and variances while a recipe is selected. It can be used when looking up data directly from the database.	String Read Only
RecipeVariancesExists	If true, then Ignition tags associated with at least one recipe value for this production item have changed.	Boolean Read Only
RecipeWriteError	If true, then at least one recipe value did not write to the associated Ignition tags when the recipe was first selected.	Boolean Read Only
ValueMonitorEnabled	If true, recipe values are being monitored and recipe value variances will be logged.	Boolean Read Only
EnableRecipe	Set to true to allow recipes to be selected for this production item. This is useful when selecting a recipe for a production line and preventing selecting the same recipe for selected child production items.	Boolean

4.6.5 Cell

Description

The cell folder contains some properties associated with the production cell. The name is the same as the cell name that is configured in the designer. The image below represents the **Filler** of the OEEDemo project.



Cell

Child Folders

RecipeValue Any recipe values that are configured for the production cell will appear in this folder.

Properties

ActiveRecipeName	If a recipe is active for this production cell, then this is the name of the recipe. If a recipe is not active, then this is blank.	String Read Only
RecipeLoading	True if a recipe is currently being loaded for the production cell.	Boolean Read Only
RecipeActive	Indicates if a recipe is currently active.	Boolean Read Only
RecipeScale	Set this to the amount to scale a recipe prior to selecting a recipe for the production cell.	Double
RecipeTrackingUUID	This is a unique value used for tracking initial recipe values and variances while a recipe is selected. It can be used when looking up data directly from the database.	String Read Only

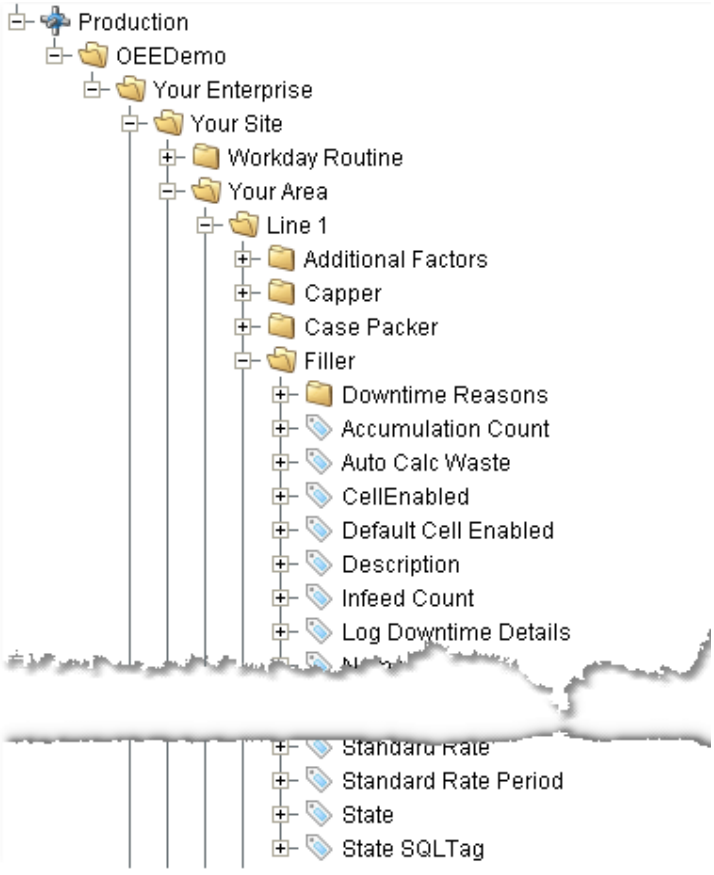
© Inductive Automation

RecipeVariancesExists	If true, then Ignition tags associated with at least one recipe value for this production item have changed.	Boolean Read Only
RecipeWriteError	If true, then at least one recipe value did not write to the associated Ignition tags when the recipe was first selected.	Boolean Read Only
ValueMonitorEnabled	If true, recipe values are being monitored and recipe value variances will be logged.	Boolean Read Only
EnableRecipe	Set to true to allow recipes to be selected for this production item. This is useful when selecting a recipe for a production line and preventing selecting the same recipe for selected child production items.	Boolean

4.6.6 Cell Group

Description

The cell folder contains some properties associated with the production cell. The name is the same as the cell name that is configured in the designer. The image below represents the **Filler** of the OEEDemo project.



Cell

© Inductive Automation

Child Folders

Cell	One folder will exist for each ***Cell*** that has been configured in the Ignition Designer. The folder can be opened to view all values within the cell.
RecipeValue	Any recipe values that are configured for the production cell group will appear in this folder.

Properties

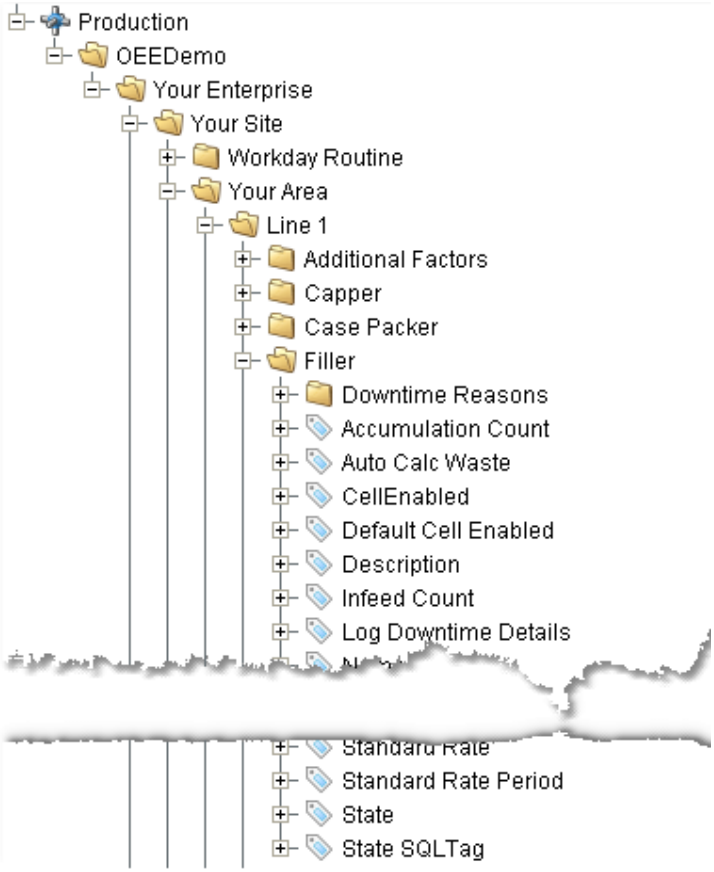
ActiveRecipeName	If a recipe is active for this production cell, then this is the name of the recipe. If a recipe is not active, then this is blank.	String Read Only
RecipeLoading	True if a recipe is currently being loaded for the production cell group.	Boolean Read Only
RecipeActive	Indicates if a recipe is currently active.	Boolean Read Only
RecipeScale	Set this to the amount to scale a recipe prior to selecting a recipe for the production cell group.	Double
RecipeTrackingUUID	This is a unique value used for tracking initial recipe values and variances while a recipe is selected. It can be used when looking up data directly from the database.	String Read Only
RecipeVariancesExists	If true, then Ignition tags associated with at least one recipe value for this production item have changed.	Boolean Read Only
RecipeWriteError	If true, then at least one recipe value did not write to the associated Ignition tags when the recipe was first selected.	Boolean Read Only
ValueMonitorEnabled	If true, recipe values are being monitored and recipe value variances will be logged.	Boolean Read Only
EnableRecipe	Set to true to allow recipes to be selected for this production item. This is useful when selecting a recipe for a production line and preventing selecting the same recipe for selected child production items.	Boolean

4.6.7 Location

Description

The cell folder contains some properties associated with the production cell. The name is the same as the cell name that is configured in the designer. The image below represents the **Filler** of the OEEDemo project.

© Inductive Automation


Production
OEEDemo
Your Enterprise
Your Site
Workday Routine
Your Area
Line 1
Additional Factors
Capper
Case Packer
Filler
Downtime Reasons
Accumulation Count
Auto Calc Waste
CellEnabled
Default Cell Enabled
Description
Infeed Count
Log Downtime Details
Standard Rate Period
State
State SQLTag

Cell

Child Folders

RecipeValue	Any recipe values that are configured for the production location will appear in this folder.

Properties

ActiveRecipeName	If a recipe is active for this production cell, then this is the name of the recipe. If a recipe is not active, then this is blank.	String Read Only
RecipeLoading	True if a recipe is currently being loaded for the production location.	Boolean Read Only
RecipeActive	Indicates if a recipe is currently active.	Boolean Read Only
RecipeScale	Set this to the amount to scale a recipe prior to selecting a recipe for the production location.	Double
RecipeTrackingUUID	This is a unique value used for tracking initial recipe values and variances while a recipe is selected. It can be used when looking up data directly from the database.	String Read Only
RecipeVariancesExists	If true, then Ignition tags associated with at least one recipe value for this production item have changed.	Boolean Read Only
RecipeWriteError	If true, then at least one recipe value did not write to the associated Ignition tags when the recipe was first selected.	Boolean Read Only
ValueMonitorEnabled	If true, recipe values are being monitored and recipe value variances will be logged.	Boolean Read Only
EnableRecipe	Set to true to allow recipes to be selected for this production item. This is useful when selecting a recipe for a production line and preventing selecting the same recipe for selected child production items.	Boolean

4.7 Scripting

This section is a reference for scripting functions provided by the Recipe Module. It also has a reference for any objects that are used by or returned by the scripting functions.

4.7.1 Client / Gateway Scripts

The Recipe / Changeover Module exposes many script functions that support managing recipes. In fact, the internal functions used by the recipe editor and other recipe components are exposed as script functions that can be used on the client or the gateway.

© Inductive Automation

4.7.1.1 getDefaultValues

```
system.recipe.getDefaultValues(projectName, itemPath, category, subProductCode)
```

Return values for a sub recipe based on a product code or default values for a production item.

This script function can be used in gateway client scripts. When called from the client, omit the projectName parameter.

parameters

`projectName`	The project name. Only include when called from the gateway. When called from the client, the project is the same as current project for the client. Data Type String
`itemPath`	The item path to a production line, cell, cell group or location. For example: "Your Enterprise\Your Site\Your Area\Line 1" Data Type String
category	Category of recipe values to return. Where 1 is recipe values created by the recipe module, 2 is recipe values created by the OEE module and 3 is recipe values created by the SPC module. Use blank to include all categories. Data Type String
`subProductCode`	Sub product code to return values for, or else leave blank to read the default values for the production item. Data Type String

returns

`List<ItemRecipeValue>`	A list of ItemRecipeValue objects. Data Type String

Example

```
itemPath = event.source.parent.getComponent('Production Line Selector').selectedPathWithoutProject

#Request the default values for the selected production item
list = system.recipe.getDefaultValues(itemPath, "1", "")
system.gui.messageBox(str(list.size()))
for rv in list:
	system.gui.messageBox("%s = %s" % (rv.getName(), str(rv.getValue())))
```

4.7.1.2 setPathDefaultValue

```
system.recipe.setPathDefaultValue(projectName, itemPath, subProductCode, valueName, value, note)
```

Set a production item sub recipe or default value.

This script function can be used in gateway client scripts. When called from the client, omit the projectName parameter.

parameters

projectName	The project name. Only include when called from the gateway. When called from the client, the project is the same as current project for the client. Data Type String
itemPath	The item path to a production line, cell, cell group or location. For example: "Your Enterprise\Your Site\Your Area\Line 1" Data Type String
subProductCode	Sub product code to set value for, or else leave blank to set the default value for the production item. Data Type String
valueName	User name for this comment Data Type String
value	Set the recipe value to this value. Data Type String
note	Optional note to be stored in the recipe change log. Data Type String

returns

none

4.7.1.3 revertPathDefaultValue

```
system.recipe.revertPathDefaultValue(projectName, itemPath, subProductCode, valueNames, note)
```

Revert production item default values back to be inherited from the parent.

This script function can be used in gateway client scripts. When called from the client, omit the projectName parameter.

parameters

projectName	The project name. Only include when called from the gateway. When called from the client, the project is the same as current project for the client. Data Type String
itemPath	The item path to a production line, cell, cell group or location. For example: "Your Enterprise\Your Site\Your Area\Line 1" Data Type String
subProductCode	Sub product code to revert value(s) for, or else leave blank to revert the default value(s) for the production item. Data Type String
valueNames	One or more recipe value names separated by commas to revert. Data Type String
note	Optional note to be stored in the recipe change log. Data Type String

returns

none

© Inductive Automation

4.7.1.4 createSubProductCode

```
system.recipe.createSubProductCode(projectName, itemPath, subProductCode, note)
```

Create a new sub product code (sub recipe).

This script function can be used in gateway client scripts. When called from the client, omit the projectName parameter.

parameters

`projectName`	The project name. Only include when called from the gateway. When called from the client, the project is the same as current project for the client. Data Type String
`itemPath`	The item path to a production line, cell, cell group or location. For example: "Your Enterprise\Your Site\Your Area\Line 1" Data Type String
`subProductCode`	New sub product code. Data Type String
`note`	Optional note to be stored in the recipe change log. Data Type String

returns

none

4.7.1.5 renameSubProductCode

```
system.recipe.renameSubProductCode(projectName, itemPath, existingSubProductCode,
newSubProductCode, note)
```

Adds a comment note to the current run for the selected line.

This script function can be used in gateway client scripts. When called from the client, omit the projectName parameter.

parameters

projectName	The project name. Only include when called from the gateway. When called from the client, the project is the same as current project for the client. Data Type String
linePath	The item path to a production line, cell, cell group or location. For example: "Your Enterprise\Your Site\Your Area\Line 1" Data Type String
existingSubProductCode	Existing sub product code name. Data Type String
newSubProductCode	New sub product code name. Data Type String
note	Optional note to be stored in the recipe change log. Data Type String

returns

none

4.7.1.6 deleteSubProductCode

```
system.recipe.deleteSubProductCode(projectName, itemPath, subProductCode, note)
```

Delete sub product code.

This script function can be used in gateway client scripts. When called from the client, omit the projectName parameter.

parameters

`projectName` — The project name. Only include when called from the gateway. When called from the client, the project is the same as current project for the client.
Data Type String

`itemPath` — The item path to a production line, cell, cell group or location.
For example: "Your Enterprise\Your Site\Your Area\Line 1"
Data Type String

`subProductCode` — Sub product code name.
Data Type String

`note` — Optional note to be stored in the recipe change log.
Data Type String

returns

none

4.7.1.7 setPathRecipeValue

```
system.recipe.setPathRecipeValue(projectName, itemPath, recipeName, valueName, value, note)
```

Set production item recipe value.

This script function can be used in gateway client scripts. When called from the client, omit the projectName parameter.

parameters

`projectName` — The project name. Only include when called from the gateway. When called from the client, the project is the same as current project for the client.
Data Type String

`itemPath` — The item path to a production line, cell, cell group or location.
For example: "Your Enterprise\Your Site\Your Area\Line 1"
Data Type String

`recipeName` — Name of recipe.
Data Type String

`valueName` — Recipe value name to set.
Data Type String

`value` — New value to assign to recipe value.
Data Type String

`note` — Optional note to be stored in the recipe change log.
Data Type String

returns

none

Example:

```
itemPath = event.source.parent.getComponent('Production Line Selector').selectedPathWithoutProject
recipe = 'Recipe C1 6Pk'
valueName = 'Line Speed'
value = '99'
system.recipe.setPathRecipeValue(itemPath, recipe, valueName, value, '')
```

© Inductive Automation

4.7.1.8 revertPathRecipeValues

```
system.recipe.revertPathRecipeValues(projectName, itemPath, recipeName, valueNames, note)
```

Revert production item recipe values back to the parent production item.

This script function can be used in gateway client scripts. When called from the client, omit the projectName parameter.

parameters

projectName	The project name. Only include when called from the gateway. When called from the client, the project is the same as current project for the client. Data Type String
itemPath	The item path to a production line, cell, cell group or location. For example: "Your Enterprise\Your Site\Your Area\Line 1" Data Type String
recipeName	Name of recipe. Data Type String
valueNames	One or more recipe value names separated by commas to revert. Data Type String
note	Optional note to be stored in the recipe change log. Data Type String

returns

none

4.7.1.9 createRecipe

```
system.recipe.createRecipe(projectName, recipeName, parentRecipeName, note)
```

Create new recipe.

This script function can be used in gateway client scripts. When called from the client, omit the projectName parameter.

parameters

`projectName` — The project name. Only include when called from the gateway. When called from the client, the project is the same as current project for the client.
Data Type String

`recipeName` — Name of new recipe.
Data Type String

`parentRecipeName` — Name of parent recipe to base this recipe. Leave blank if new recipe is not based on any other recipe.
Data Type String

`note` — Optional note to be stored in the recipe change log.
Data Type String

returns

none

4.7.1.10 renameRecipe

```
system.recipe.renameRecipe(projectName, existingRecipeName, newRecipeName, note)
```

Rename specified recipe.

This script function can be used in gateway client scripts. When called from the client, omit the projectName parameter.

parameters

projectName	The project name. Only include when called from the gateway. When called from the client, the project is the same as current project for the client. Data Type String
existingRecipeName	Name of the existing recipe. Data Type String
newRecipeName	New recipe name. Data Type String
note	Optional note to be stored in the recipe change log. Data Type String

returns

none

4.7.1.11 deleteRecipe

```
system.recipe.deleteRecipe(projectName, recipeName, note)
```

Delete specified recipe.

This script function can be used in gateway client scripts. When called from the client, omit the projectName parameter.

parameters

projectName	The project name. Only include when called from the gateway. When called from the client, the project is the same as current project for the client. Data Type String
recipeName	Name of the recipe to delete. Data Type String
note	Optional note to be stored in the recipe change log. Data Type String

returns

none

4.7.1.12 addItemToRecipe

```
system.recipe.addItemToRecipe(projectName, recipeName, itemPath, note)
```

Add a production item to a recipe. Once a production item is added to a recipe, the recipe values for the production item can be managed. Also, the recipe can be selected for the added production item.

This script function can be used in gateway client scripts. When called from the client, omit the projectName parameter.

parameters

`projectName` — The project name. Only include when called from the gateway. When called from the client, the project is the same as current project for the client.
Data Type String

`recipeName` — Name of the recipe to add the specified production item.
Data Type String

`itemPath` — The item path to the production line, cell, cell group or location to add to the recipe.
For example: "Your Enterprise\Your Site\Your Area\Line 1"
Data Type String

`note` — Optional note to be stored in the recipe change log.
Data Type String

returns

none

4.7.1.13 removeItemFromRecipe

```
system.recipe.removeItemFromRecipe(projectName, recipeName, itemPath, note)
```

Remove a production item from a recipe.

This script function can be used in gateway client scripts. When called from the client, omit the projectName parameter.

parameters

projectName	The project name. Only include when called from the gateway. When called from the client, the project is the same as current project for the client. Data Type String
recipeName	Name of the recipe to remove the specified production item. Data Type String
linePath	The item path to the production line, cell, cell group or location to remove from the recipe. For example: "Your Enterprise\Your Site\Your Area\Line 1" Data Type String
note	Optional note to be stored in the recipe change log. Data Type String

returns

none

4.7.1.14 getChangelogHistory

```
system.recipe.getChangelogHistory(projectName, changelogFilters)
```

Based on the filters set in the changelogFilters parameter, return change log history for recipe. See Recipe Change Log for more information.

This script function can be used in gateway client scripts. When called from the client, omit the projectName parameter.

parameters

`projectName` The project name. Only include when called from the gateway. When called from the client, the project is the same as current project for the client.
Data Type String

`changelogFilters` Change log filters (See ChangelogFilters object for more information).
Data Type ChangelogFilters

returns A Dataset object containing rows and columns of change log history.
`Dataset`

Example

```
#Collect values we want to filter by
projectName = system.util.getProjectName()
itemPath = event.source.parent.getComponent('Production Line Selector').selectedPathWithoutProject
fromDate = event.source.parent.getComponent('Date Range').startDate
toDate = event.source.parent.getComponent('Date Range').endDate

#Build the filters object
filters = system.recipe.filter.changelog.createNew()
filters.setProjectName(projectName)
filters.addCategory("Recipe")
filters.setItemPathFilter(itemPath)
filters.setFromDate(fromDate)
filters.setToDate(toDate)

#Request the change log for the given filters
ds = system.recipe.getChangelogHistory(filters)
event.source.parent.getComponent('Table').data = ds
```

4.7.1.15 getRecipeVariances

```
system.recipe.getRecipeVariances(projectName, varianceFilters)
```

Based on the filters set in the varianceFilters parameter, return recipe value variances. See Variance Monitoring for more information.

This script function can be used in gateway client scripts. When called from the client, omit the projectName parameter.

parameters

`projectName`	The project name. Only include when called from the gateway. When called from the client, the project is the same as current project for the client. Data Type String
`varianceFilters`	Change log filters (See VarianceFilters object for more information). Data Type VarianceFilters

returns A Dataset object containing rows and columns of recipe value variances.
`Dataset`

Example

```
#Collection values we want to filter by
projectName = system.util.getProjectName()
itemPath = event.source.parent.getComponent('Production Line Selector').selectedPathWithoutProject

#Build the filters object
filters = system.recipe.filter.variance.createNew()
filters.setProjectName(projectName)
filters.setVarianceEntryType("Recipe")
filters.setVarianceScopeTypes("Last")
filters.setItemPath(itemPath)

#Request the variances for the given filters
ds = system.recipe.getRecipeVariances(filters)
event.source.parent.getComponent('Table').data = ds
```

4.7.1.16 getItemRecipeList

```
system.recipe.getItemRecipeList(projectName, itemPath, recipeFilter)
```

Return the current recipes available for a production item.

This script function can be used in gateway client scripts. When called from the client, omit the projectName parameter.

parameters

projectName	The project name. Only include when called from the gateway. When called from the client, the project is the same as current project for the client. Data Type String
itemPath	The item path to a production line, cell, cell group or location. For example: "Your Enterprise\Your Site\Your Area\Line 1" Data Type String
recipeFilter	Optional recipe filter. The filter can contain ? and * wild card characters. Data Type String

returns

List<String>

Example

```
itemPath = event.source.parent.getComponent('Production Line Selector').selectedPathWithoutProject

#Request the available recipes for the selected production item.
list = system.recipe.getItemRecipeList(itemPath, "Recipe C*")
for recipeName in list:
	system.gui.messageBox(recipeName)
```

© Inductive Automation

4.7.1.17 getCurrentItemRecipe

```
system.recipe.getCurrentItemRecipe(projectName, itemPath)
```

Return the current selected recipe name for a production item.

This script function can be used in gateway client scripts. When called from the client, omit the projectName parameter.

parameters

`projectName`	The project name. Only include when called from the gateway. When called from the client, the project is the same as current project for the client. `Data Type` `String`
`itemPath`	The item path to a production line, cell, cell group or location. For example: "Your Enterprise\Your Site\Your Area\Line 1" `Data Type` `String`

returns Current selected recipe name of specified production item.
`String`

4.7.1.18 getRecipeValues

```
system.recipe.getRecipeValues(projectName, itemPath, recipeName, category)
```

Return recipe values for a production item and recipe combination.

This script function can be used in gateway client scripts. When called from the client, omit the projectName parameter.

parameters

`projectName`	The project name. Only include when called from the gateway. When called from the client, the project is the same as current project for the client. Data Type String
`itemPath`	The item path to a production line, cell, cell group or location. For example: "Your Enterprise\Your Site\Your Area\Line 1" Data Type String
`recipeName`	The recipe name. Data Type String
`category`	Category of recipe values to return. Where 1 is recipe values created by the recipe module, 2 is recipe values created by the OEE module and 3 is recipe values created by the SPC module. Use blank to include all categories. Data Type String

returns A list of ItemRecipeValue objects (See ItemRecipeValue object for more information).
Data Type `List<ItemRecipeValue>`

Example

```
itemPath = event.source.parent.getComponent('Production Line Selector').selectedPathWithoutProject

#Request the recipe values for production item and recipe combination.
list = system.recipe.getRecipeValues(itemPath, "Recipe C1 6Pk", "1")
system.gui.messageBox(str(list.size()))
for rv in list:
	system.gui.messageBox("%s = %s" % (rv.getName(), str(rv.getValue())))
```

4.7.1.19 isItemRecipeMonitoringEnabled

```
system.recipe.isItemRecipeMonitoringEnabled(projectName, itemPath)
```

Return recipe value monitoring enabled state. See Variance Monitoring for more information.

This script function can be used in gateway client scripts. When called from the client, omit the projectName parameter.

parameters

`projectName` — The project name. Only include when called from the gateway. When called from the client, the project is the same as current project for the client.
Data Type String

`itemPath` — The item path to a production line, cell, cell group or location.
For example: "Your Enterprise\Your Site\Your Area\Line 1"
Data Type String

returns True, if recipe value monitoring is enabled.
`Boolean`

4.7.1.20 setItemRecipe

```
system.recipe.setItemRecipe(projectName, itemPath, recipeName, enableValueMonitoring)
```

Set the recipe for the production item specified by the itemPath parameter. If the production item is a line, then all children production items of the line will also be set to the same recipe provided they were added to the recipe.

This script function can be used in gateway client scripts. When called from the client, omit the projectName parameter.

parameters

`projectName` — The project name. Only include when called from the gateway. When called from the client, the project is the same as current project for the client.
Data Type String

`itemPath` — The item path to a production line, cell, cell group or location. For example: "Your Enterprise\Your Site\Your Area\Line 1"
Data Type String

`recipeName` — The name of the recipe.
Data Type String

`enableValueMonitoring` — If true, turn on recipe value variance monitoring. See Variance Monitoring for more information.
Data Type Boolean

returns

none

4.7.1.21 cancelItemRecipe

```
system.recipe.cancelItemRecipe(projectName, itemPath)
```

Cancel the current recipe for the production item specified by the itemPath parameter. If the production item is a line, then the recipe for all children production items of the line will also be canceled.

This script function can be used in gateway client scripts. When called from the client, omit the projectName parameter.

parameters

`projectName`	The project name. Only include when called from the gateway. When called from the client, the project is the same as current project for the client. `Data Type` `String`
`itemPath`	The item path to a production line, cell, cell group or location. For example: "Your Enterprise\Your Site\Your Area\Line 1" `Data Type` `String`

returns

none

4.7.1.22 readItemCurrentValues

```
system.recipe.readItemCurrentValues(projectName, itemPath, includeChildren, recipeName,
subProductCode, valueNames, note)
```

Set the recipe values to the current tag value(s) for the production item specified by the itemPath parameter.

This script function can be used in gateway client scripts. When called from the client, omit the projectName parameter.

parameter

`projectName` — The project name. Only include when called from the gateway. When called from the client, the project is the same as current project for the client.
Data Type String

`itemPath` — The item path to a production line, cell, cell group or location. For example: "Your Enterprise\Your Site\Your Area\Line 1"
Data Type String

`includeChildren` — If true, also set the recipe values for children of the items.
Data Type Boolean

`recipeName` — The name of the recipe (only applies for recipes).
Data Type String

`subProductCode` — The sub product code (only applies for sub recipes).
Data Type String

`valueNames` — The name(s) of the recipe values to set to current tag values. Separate multiple recipe value names with commas. For all recipe values of a production item, leave blank.
Data Type String

`note` — Optional note to be stored in the recipe change log.
Data Type String

returns

`none`

4.7.1.23 exportRecipe

```
system.recipe.exportRecipe(projectName, filters)
```

Adds a comment note to the current run for the selected line. See the Import / Export section of Editing Recipes for CSV file format and other information.

This script function can be used in gateway client scripts. When called from the client, omit the projectName parameter.

parameters

`projectName`	The project name. Only include when called from the gateway. When called from the client, the project is the same as current project for the client. `Data Type` `String`
`filters`	Filter statements separated by commas. See the Recipe Analysis Provider for more information on the available filters. `Data Type` `String`

returns CSV formatted string containing the recipe values.
`String`

Example

```
itemPath = event.source.parent.getComponent('Production Line Selector').selectedLinePath
filters = "Children=Include,Recipe Name=Master C,Item Path=%s" % (itemPath)
csv = system.recipe.exportRecipe(filters)
system.file.writeFile("C:\\Temp\\recipe_export.csv", csv, False)
```

© Inductive Automation

4.7.1.24 importRecipe

```
system.recipe.importRecipe(projectName, csvData, note)
```

Set the recipe values to the current tag value(s) for the production item specified by the itemPath parameter. See the Import / Export section of Editing Recipes for CSV file format and other information.

Values that are outside of the range defined in the recipe values security will not be imported. When this happens, an exception is returned listing all of the values that were not imported. See Recipe Security for more information.

This script function can be used in gateway client scripts. When called from the client, omit the projectName parameter.

parameters

`projectName`	The project name. Only include when called from the gateway. When called from the client, the project is the same as current project for the client. Data Type String
`csvData`	String in CSV format containing recipe values. Must be the same format that is returned by the exportRecipe function. Data Type String
`note`	Optional note to be stored in the recipe change log. Data Type String

returns

none

Example

```
csv = system.file.readFileAsString("C:\\Temp\\recipe_export.csv")
itemPath = event.source.parent.getComponent('Production Line Selector').selectedLinePath
system.recipe.importRecipe(csv, "Values set during import.")
```

4.7.2 Recipe Value Scripts

When recipe values are added to a production item, there are properties that allow script to be entered. The following sections detail the different events and how they are used. See Adding a Recipe Value for more information.

4.7.2.1 Evaluate Variance Script

If the Evaluate Variance Script property (see Adding a Recipe Value for more information) of a production item contains a script, it will be executed every time an Ignition tag associated with a recipe value changes. Variance monitoring must also be active. When it is executed, the event object provides information about the recipe value being evaluated. If event.setLogVariance(logVariance) is not called, then the log variance state determined before this event will be used. See Variance Monitoring for more information.

Event Properties

event.getRecipeTag()	Returns RecipeTag object associated with recipe value. The

RecipeTag object contains details about the recipe value and associated Ignition tag. (See RecipeTag object for more information).
Data Type RecipeTag

event.getScale() Returns the recipe scale being used when the recipe was selected.
Data Type Double

event.isLogVariance() Returns the current log variance state for the recipe value. If true, a variance will be logged.
Data Type Boolean

event.setLogVariance(logVariance)
parameters
logVariance If true a variance will be logged.
Data Type Boolean
returns
none

Example

```
upperLimit = system.tag.read("[Default]UpperLimit")
lowerLimit = system.tag.read("[Default]LowerLimit")
recipeValue = event.getRecipeTag().getCurrentValue()
if recipeValue > upperLimit.value or recipeValue < lowerLimit.value:
	event.setLogVariance(True)
else:
	event.setLogVariance(False)
```

4.7.2.2 Request Value Script

If the Request Value Script property (see Adding a Recipe Value for more information) of a production item contains a script, it will be executed every time a recipe is selected. When it is executed, the event object provides information about the recipe value being read. If event.setRecipeValue(value) is not called, then the value from the database will be used.

Event Properties

event.getItemPath() Returns the production item path that recipe value belongs to.
Data Type String

event.getValueName() Returns the name of the recipe value being read.
Data Type String

event.getTagPath() Returns the Ignition tag path associated with the recipe value.
Data Type String

event.getScale() Returns the recipe scale being used while reading the current recipe.
Data Type Double

event.getRecipeValue() Returns the recipe value from the database. This is the value that was set in the recipe editor or from a script function.
Data Type String

event.setRecipeValue(value) **parameters**

value	Value to change the current recipe value.	
Data Type	Data Type	String

returns

none

Example

```
import math
try:
       value = float(event.getRecipeValue())
       event.setRecipeValue(str(math.log10(value)))
except:
       event.setRecipeValue("0")
```

4.7.3 Object Reference

The Recipe Module has script functions and events that use various objects. This is because some recipe information contains more data than can be represented with a single primitive data type. For example a recipe value has a name, description, units, format and more, and the ItemRecipeValue is used to hold all of this information when returning back recipe value information from a script function.

The following sections provide documentation of the methods and properties associated with these various objects.

4.7.3.1 ChangelogFilters

A ChangeLogFilters object is used when requesting recipe change logs with the getChangelogHistory script function to narrow down the results that are returned. For example, if you only want the change log history for a specific production item (machine) and specific date range, the ChangeLogFilters object properties are set appropriately and are passed as parameters to the getChangelogHistory script function.

methods:

- createNew() - ChangeLogFilters
  Returns a new instance of a ChangeLogFilters object. After setting various filter properties, it is used with the getChangelogHistory script function.

properties:

- setProjectName(String projectName)
  Set the project name to read recipe change log history. Recipe change log history is kept by project, and the project name is required with the getChangelogHistory script function.

- addCategory(String category)
  Add a category to include in the recipe change log history results. Multiple categories can be specified to be included in the results. Valid values are
  - ❑ RECIPE to include change log entries dealing with recipe changes. Recipe changes include, adding new recipes, renaming recipes, deleting recipes, adding production items to recipes, etc.
  - ❑ RECIPE_VALUE to include change log entries dealing with changes of recipe values. This includes changing a value, reverting a value back to be inherited, etc.
  - ❑ SUB_PRODUCT_CODE to include change log entries dealing with sub product codes. This includes adding new sub product codes, renaming sub product codes, deleting sub product codes, etc.

© Inductive Automation

- ❑ SUB_PRODUCT_CODE_VALUE to include change log entries dealing with changes of sub product code values or default values for a production item. This includes changing a value, reverting a value back to be inherited, etc.

- removeCategory(String category)
  Remove a category for what has already been added.

- setFromDate(Date fromDate)
  Set the start of the date range to return change log history for.

- setToDate(Date toDate)
  Set the end of the date range to return change log history for.

- setItemPathFilter(String itemPath)
  Set the path of the production item to return change log history for.
  For example: "Your Enterprise\Your Site\Your Area\Line 1"

- setRecipeNameFilter(String recipeName)
  Set an optional recipe filter. The filter can contain ? and * wild card characters.
  For example: "Recipe C*" will include all recipes that start with Recipe C. Recipe C1 and Recipe C21 will be included but Recipe D1 will not.

- setSubProductCodeFilter(String subProductCode)
  Set an optional sub product code filter. The filter can contain ? and * wild card characters.

- setUserFilter(String userName)
  Set an optional user name filter. The filter can contain ? and * wild card characters.

- setValueNameFilter(String recipeValueName)
  Set an optional recipe value name filter. The filter can contain ? and * wild card characters.

Example

```
#Collect values we want to filter by
projectName = system.util.getProjectName()
itemPath = event.source.parent.getComponent('Production Line Selector').selectedPathWithoutProject
fromDate = event.source.parent.getComponent('Date Range').startDate
toDate = event.source.parent.getComponent('Date Range').endDate

#Build the filters object
filters = system.recipe.filter.changelog.createNew()
filters.setProjectName(projectName)
filters.addCategory("Recipe")
filters.setItemPathFilter(itemPath)
filters.setFromDate(fromDate)
filters.setToDate(toDate)

#Request the change log for the given filters
ds = system.recipe.getChangelogHistory(filters)
event.source.parent.getComponent('Table').data = ds
```

© Inductive Automation

4.7.3.2 VarianceFilters

A VarianceFilters object is used when requesting variances with the getRecipeVariances script function to narrow down the results that are returned. For example, if you only want variances for a specific production item (machine) and specific date range, the VarianceFilters object properties are set appropriately and are passed as parameters to the getRecipeVariances script function.

methods:

- createNew() - VarianceFilters
  Returns a new instance of a VarianceFilters object. After setting various filter properties, it is used with the getRecipeVariances script function.

properties:

- setProjectName(String projectName)
  Set the project name to read variances. Variances are kept by project, and the project name is required with the getRecipeVariances script function.

- setVarianceEntryType(String varianceType)
  Set the variance types to include in the results. Valid values are
  - ❑ RECIPE to return variances that occurred while a production item was selected to a recipe.
  - ❑ SUB_RECIPE to return variances that occurred while a sub product code was selected for a production item. See Sub Recipes for more information.

- setVarianceScopeTypes(String varianceScopeType)
  Set the variance scope to include in the results. Valid values are
  - ❑ LAST to return variances that occurred for the current or last recipe that a production item was set. This is useful for detecting any variances in real time for a production run. If the production run has stopped, it will return the variances as long as a new recipe has not been selected for the production item.
  - ❑ DATE_RANGE to return variances for the date range specified with the setFromDate() and setToDate() properties.

- setFromDate(Date fromDate)
  Set the start of the date range to return variances for if setVarianceScopeTypes ("DATE_RANGE") is called.

- setToDate(Date toDate)
  Set the end of the date range to return variances for if setVarianceScopeTypes ("DATE_RANGE") is called.

- setItemPath(String itemPath)
  Set the path of the production item to return variances for.
  For example: "Your Enterprise\Your Site\Your Area\Line 1"

- setIncludeChildren(Boolean includeChildren)
  Set if children production items under the production item specified by the setItemPath() property, should be included in the variance results.

- `setIncludeInitialValues`(Boolean includeInitialValues)

© Inductive Automation

Set if the initial values (meaning the values when the recipe was first selected) should be included in the variance results.

- `setIncludeVarianceValues`(Boolean includeVarianceValues)
  Set if the variance values (meaning the values that changed after the recipe was first selected) should be included in the variance results.

- `setRecipe`(String recipeName)
  Set an optional recipe filter. The filter can contain ? and * wild card characters.
  For example: "Recipe C*" will include all recipes that start with Recipe C. Recipe C1 and Recipe C21 will be included but Recipe D1 will not.

- `setSubRecipe`(String subRecipeName)
  Set an optional sub recipe filter. The filter can contain ? and * wild card characters. See Sub Recipes for more information.

- `setRecipeValueName`(String recipeValueName)
  Set an optional recipe value name filter. The filter can contain ? and * wild card characters.

Example

```
#Collection values we want to filter by
projectName = system.util.getProjectName()
itemPath = event.source.parent.getComponent('Production Line Selector').selectedPathWithoutProject

#Build the filters object
filters = system.recipe.filter.variance.createNew()
filters.setProjectName(projectName)
filters.setVarianceEntryType("Recipe")
filters.setVarianceScopeTypes("Last")
filters.setItemPath(itemPath)
filters.setIncludeChildren(False)

#Request the variances for the given filters
ds = system.recipe.getRecipeVariances(filters)
event.source.parent.getComponent('Table').data = ds
```

4.7.3.3 ItemRecipeValue

A ItemRecipeValue object is returned by many of the recipe script methods, usually as a list of ItemRecipeValue. Because a recipe value has several properties such as the name, minimum value, maximum value, units, etc., the details are returned in this ItemRecipeValue object.

Recipe values also can have varying data types and an ItemRecipeValue object supports reading the value in its true data type. For example, if a recipe value if of type Float8, then getValue(), getMinValue() and getMaxValue() all return Float8 values. This cannot be done by returning the recipe values in a Dataset where each column must be a single data type.

properties:

- getName() - String
  Returns the name of the recipe value. This is the same name entered in the recipe value entry in the designer.

© Inductive Automation

- hasDescription() - Boolean
  Returns True if a description exists for the recipe value.

- getDescription - String
  Returns the description of the recipe value. This is the description entered in the recipe value entry in the designer.

- hasDataType() - Boolean
  Returns True if the recipe value has a data type assigned. Recipe values that do not have a tag assigned to them will not have a data type. This is because the data type is obtained from the tag.

- getDataType() - DataType
  Returns the Ignition DataType for the recipe value.

- isValid() - Boolean
  Returns True if the value is valid for the data type of the recipe value.

- getValue() - Object
  Returns the value of the recipe value. The data type will be one defined by the Ignition DataType. If a value has not been assigned to the recipe value, then None will be returned and isValid() will return false.

- getMinValue() - Object
  Returns the minimum value that the value can be as defined in the Ignition tag. This is different from the recipe value security that depends on the authenticated user. See Recipe Security for more information.

- getMaxValue() - Object
  Returns the maximum value that the value can be as defined in the Ignition tag. This is different from the recipe value security that depends on the authenticated user. See Recipe Security for more information.

- getUnits() - Object
  Returns the units as defined in the Ignition tag.

- getFormat() - Object
  Returns the format as defined in the Ignition tag.

- getAssignedBy() - Object
  Returns the recipe or production item default value where the recipe value is inherited from. If the recipe value is not inherited, it will be the name of the recipe where it was overridden.

- getCategory() - Object
  Returns the MES module that created the recipe value. Where 1 is recipe value created by the recipe module, 2 is recipe value created by the OEE module and 3 is recipe value created by the SPC module.

Example

```
itemPath = event.source.parent.getComponent('Production Line Selector').
```

```
selectedPathWithoutProject
valueList = system.recipe.getDefaultValueItems(itemPath, "")
if valueList != None:
      system.gui.messageBox("Count = %d" % valueList.size())
      for itemRecipeValue in valueList:
            system.gui.messageBox("%s = %s" % (itemRecipeValue.getName(),
itemRecipeValue.getValue()))
```

4.7.3.4 RecipeTag

A RecipeTag object contains details about a recipe value. It reflects the properties that are configured in the designer when the recipe value was added plus some live information such as the current value. See Adding a Recipe Value for more information.

properties:

- getRecipeValueName() - String
  Returns the name of the recipe value. This is the same name entered in the recipe value entry in the designer.

- getTagPath() - String
  Returns the Ignition tag path assigned to the recipe value. This is the tag path entered for the recipe value.

- hasHighVarianceThresholdStatement() - Boolean
  Returns True if a high variance threshold statement was entered for the recipe value.

- getHighVarianceThresholdStatement() - String
  Returns the high variance threshold statement that was entered for the recipe value entry.

- getHighVarianceThresholdValue() - String
  Returns the high variance threshold value. This is calculated after the tag value change is detected and is used for default handling for the log variance state. It is provided here as a convenience.

- hasLowVarianceThresholdStatement() - Boolean
  Returns True if a low variance threshold statement was entered for the recipe value.

- getLowVarianceThresholdStatement() - String
  Returns the low variance threshold statement that was entered for the recipe value entry.

- getLowVarianceThresholdValue() - String
  Returns the low variance threshold value. This is calculated after the tag value change is detected and is used for default handling for the log variance state. It is provided here as a convenience.

- hasPreviousValue() - Boolean
  Returns True if a previous value has been recorded for the recipe value.

© Inductive Automation

- getPreviousValue() - Object
  Returns the previous value of the recipe value. Anytime the value of a tag associated with a recipe value changes, the previous value is saved internally. This is used for the changed from information in variance logging.

- hasCurrentValue() - Boolean
  Returns True if the recipe value has a value.

- getCurrentValue() - Object
  Returns the tag value of a tag associated with a recipe value.

- isVarianceMonitorEnabled() - Boolean
  Returns True if variance monitoring is enabled for the recipe value.

- getDataType() - Object
  Returns the data type of the tag associated with the recipe value.

- getRecipeValue() - Object
  Returns the recipe value. This can be the value that was entered in recipe editor, set using script or inherited from a parent.

- getRecipeValue(Double scale) - Object
  Returns the recipe value adjusted by the scale parameter.

- convertValue(String value) - Object
  Returns the value passed in the parameter in the correct data type for the recipe value. If the recipe value is an Int4, then the string value passed in the value parameter will be converted to an Int4 data type and returned.

- convertAndScaleValue(String value, Double scaleFactor) - Object
  Returns the value passed in the parameter in the correct data type for the recipe value with scaling.

- scaleValue(Object value, Double scaleFactor) - Object
  Scales and returns the value passed in the parameter in the same data type as the value parameter.

Example

```
upperLimit = system.tag.read("[Default]UpperLimit")
lowerLimit = system.tag.read("[Default]LowerLimit")
recipeValue = event.getRecipeTag().getCurrentValue()

rt = event.getRecipeTag()
uLimit = rt.scaleValue(upperLimit.value, event.getScale())
lLimit = rt.scaleValue(lowerLimit.value, event.getScale())

if recipeValue > uLimit or recipeValue < lLimit:
	event.setLogVariance(True)
else:
	event.setLogVariance(False)
```

Instrument Interface Module

Part V

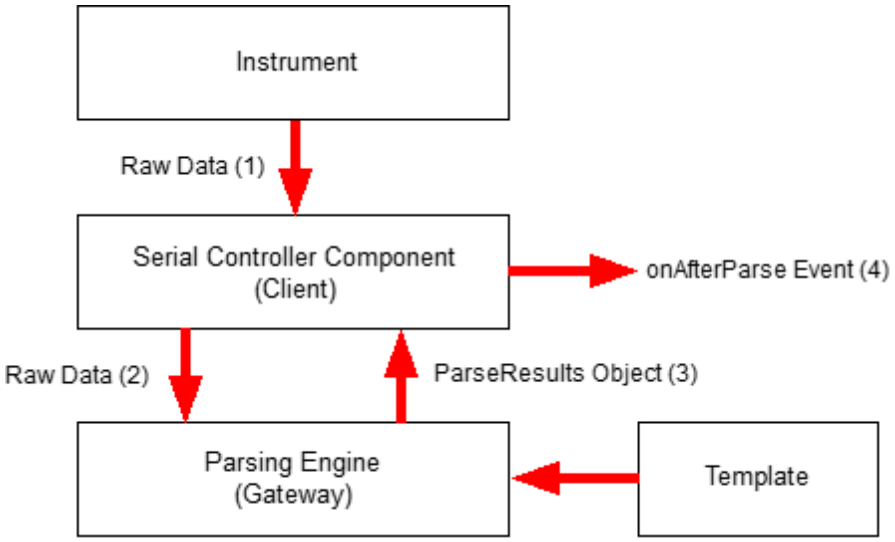
inductive automation
800.266.7798
www.inductiveautomation.com

5 Instrument Interface Module

5.1 Introduction

The Instrument Interface module is used to define communication settings and data parsing templates to an instrument. These settings and parsing templates are then used to read data from a instrument and parse the raw data to extract desired values. The data from an instrument can come from a file, serial communication port, TCP or UDP connection, OPC device such as a PLC, external data or web service.

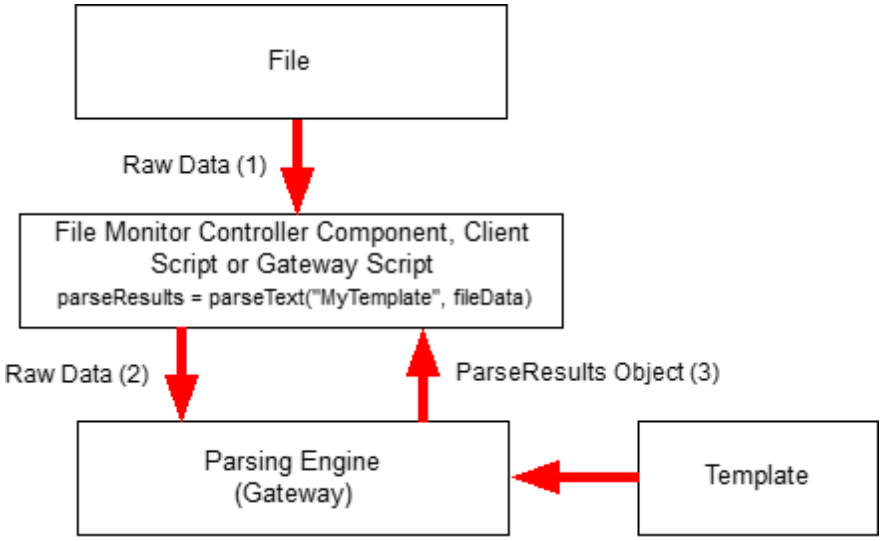
The image below show the typical flow of data when reading instrument values through a serial communications port. Note that the Client Serial Support Module is required to read serial data on a client computer. The Instrument Interface Module includes a component to make configuring and control of serial port communications easier than using the script only support of the Client Serial Support Module. If reading data from a serial communication port on the Ignition server is needed, then the Serial Server Support Module is needed.



Typical Serial Communications Flow

Some Instruments write their results to a disk file. The image below shows the typical flow when reading data from a file and using the File Monitor component or parsing script functions that are available on both the client and the gateway to parse the raw data in the file.

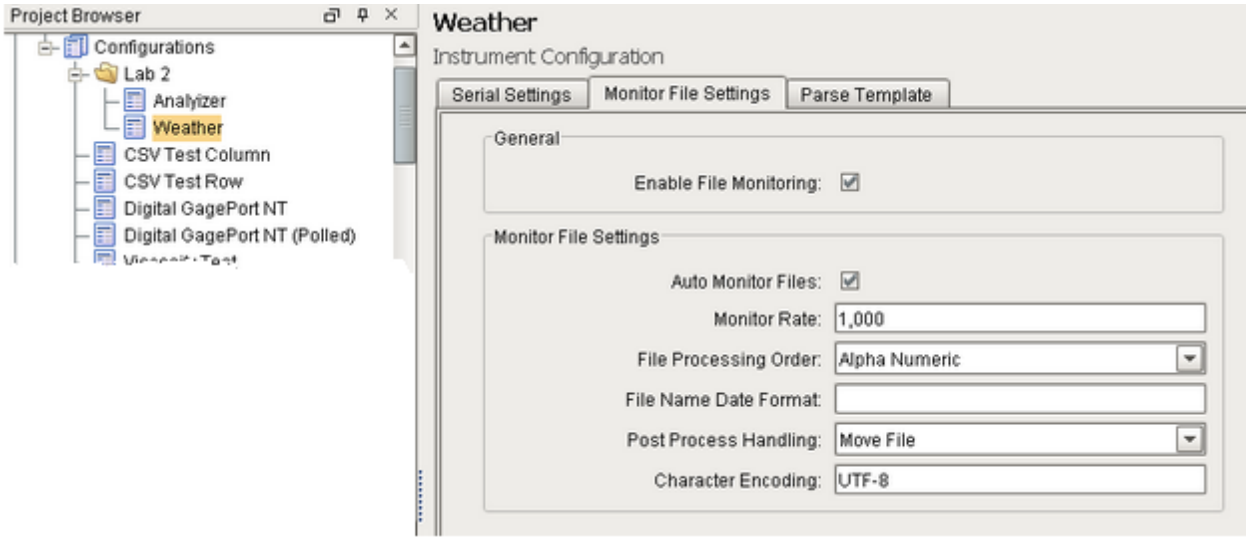
© Inductive Automation



Typical File Flow

5.2 File Monitor Settings

This page configures the file monitor settings of this Instrument Interface. It provides a configuration area by instrument type that use file method of handing off data.



File Monitor Settings

General

Enable File Monitoring If checked, these file monitoring settings will be applied to File Monitor component when this Instrument Interface is assigned to its Instrument Interface Name property.

File Monitor Settings

© Inductive Automation

Auto Monitor If true automatically detects and processes file(s) contained with the File Path property of the File Monitor component. If false, the read() of the component must be called to process file(s).

Monitoring The milliseconds between each check for new files. Any files that are found during a check will be processed. Processing of file will not overlap. If the time it takes to process the files exceeds the value of this property, then the next check will be at the next interval.

File This property defines the priority to process multiple file. It is inapplicable when a single file is selected in the File Path property. If Alpha Numeric is selected, the files are processed in alphabetical order. If Date is selected, the file names are converted to date values using the pattern defined in the File Name Date Format property and then processed in chronological order. If File Timestamp is selected, the files are processed in chronological order of the file modified date. Select from the following:

Alpha Numeric = 0
Date = 1
File Timestamp = 2

File Name This property is only applicable if the File Processing Priority property is set to Date. This property defines the parsing pattern to use when converting the file name to a date value when determining the processing order of the files. The patterns can contain both date and time format designators. See the File Name Date Format property description of the File Monitor component for more details.

© Inductive Automation

After This setting defines how files are handled after processing them. Select from the following:

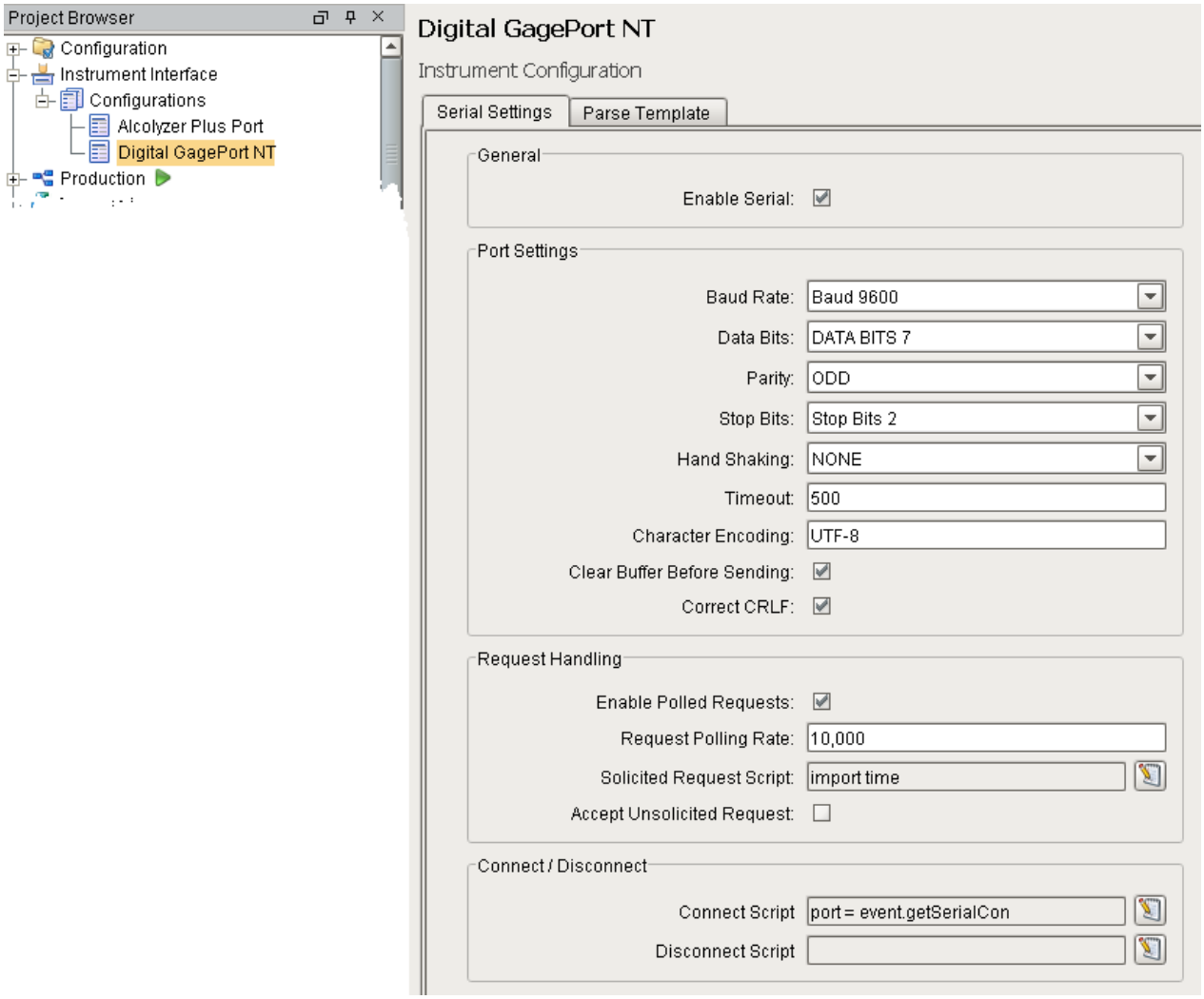
Delete File = 0
Move File = 1

Character Encoding Character encoding of the data

5.3 Serial Settings

This page configures the serial port communications settings of this Instrument Interface.

© Inductive Automation



Serial Settings Configuration

General

Enable Serial — If checked, these port settings will be applied to the Serial Controller component when this Instrument Interface is assigned to its Instrument Interface Name property.

Port Settings

Baud Rate — Serial Communication baud rate. Select from the following:

- Baud 110
- Baud 150
- Baud 300
- Baud 600
- Baud 1200
- Baud 2400
- Baud 4800
- Baud 9600
- Baud 19200
- Baud 38400

© Inductive Automation

Baud 57600
Baud 115200
Baud 230400
Baud 460800
Baud 921600

Data Bits Serial communication data bits. Select from the following:
DATA BITS 5
DATA BITS 6
DATA BITS 7
DATA BITS 8

Parity Serial communication parity. Select from the following:
NONE
EVEN
ODD
MARK
SPACE

Stop Bits Serial communication number of stop bits. Select from the following:
Stop Bits 1
Stop Bits 2

Hand Shaking Serial communication flow control methods. Select from the following:
NONE
CTS DTR
CTS RTS
DSR DTR
XON XOFF

Timeout The default number of milliseconds to wait while reading data.

Character Encoding Character encoding of the data

Clear Buffer Before Sending If checked, clears the receive buffer before sending data.

Correct CRLF If checked, corrects any combination of end of line characters to carriage return (CR) and line feed (LF).

Request Handling

Enable Polled Requests If checked, the port will be polled at the requested rate.

Request Polling Rate The rate in milliseconds to poll the port

Solicited Request Script The script to run for each polled request.
When writing scripts you can use the "event" object to reference methods in the Serial Controller component that this Instrument Interface is assigned.

Example:

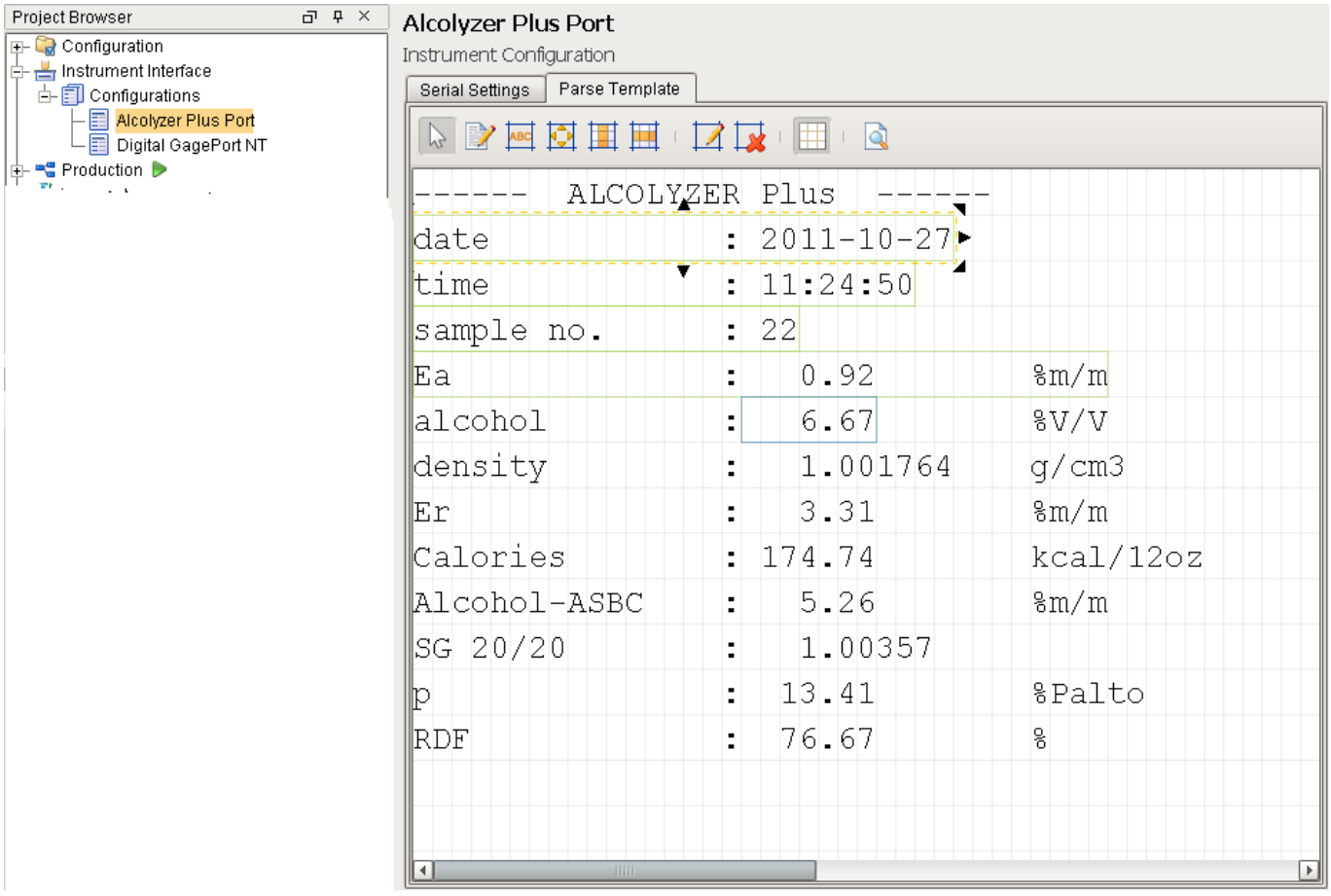
```
import time
port = event.getSerialController()
port.clearBuffer()
port.writeString("Ar")
time.sleep(0.5)
port.writeString("As")
time.sleep(0.5)
event.setReceivedData(port.readString())
```

© Inductive Automation

Accept Unsolicited Request If checked, the port will can accept requests without being solicited

5.4 Parse Template

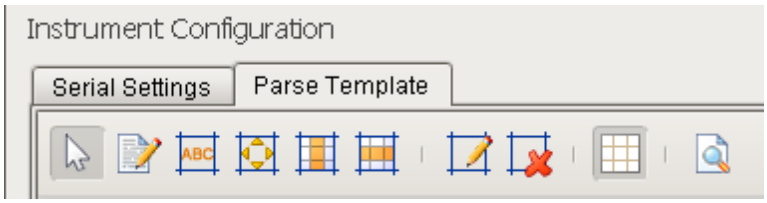
This page configures a parsing template of this instrument interface configuration. It allows a visual way to define the individual data points to extract from the raw text returned from the instrument. The text represents what is returned from a instrument and is displayed in a fixed character width. Multiple parsing boxes can then be added to define areas to extract meaningful values from.



Parse Template configuration screen

Parse Template Tools

The Parse Template configuration screen contains a toolbar palette with tools that allow interaction with the current parse template..



Parse Template tool palette

Parsing Box Selector

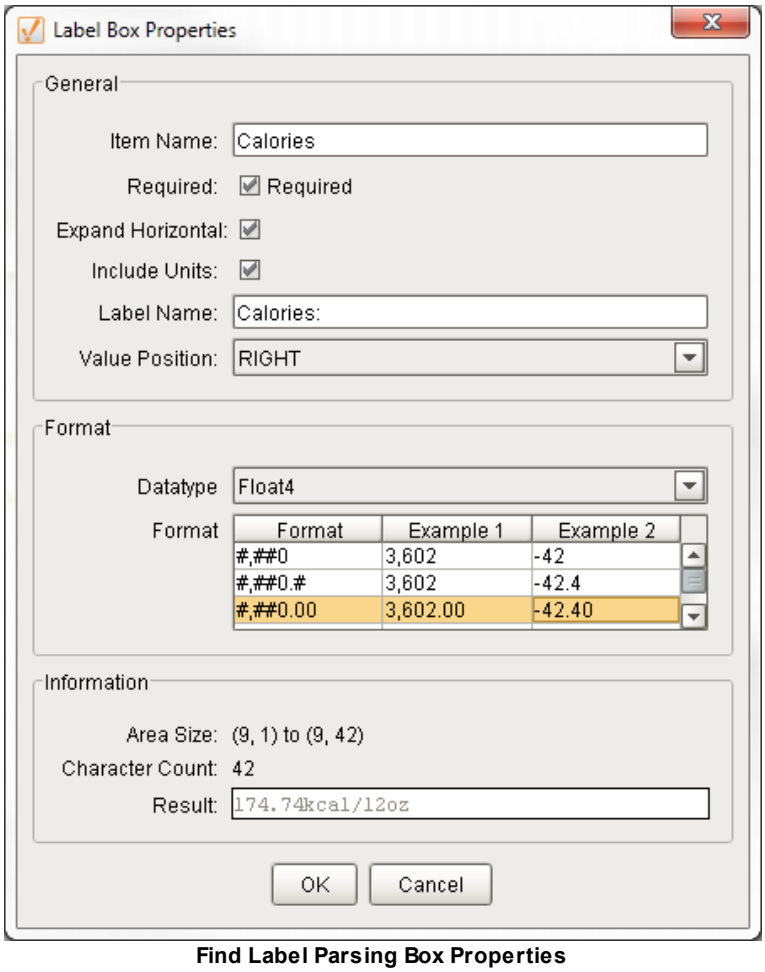
Allows the user to select any existing parsing boxes in the template. The selected parsing box will be displayed with sizing arrows on all corners.

Edit Content Text

Allows editing of the actual template text that the parsing operations will be apply to. New templates are blank and the user will need to add text representing the output received from the instrument here so that parsing boxes can be applied. This can also be populated by using the "Send to Template" menu option of the Serial Controller and File Monitor Components. Refer to the the documentation for these components for more information.

Find Label Parsing Box

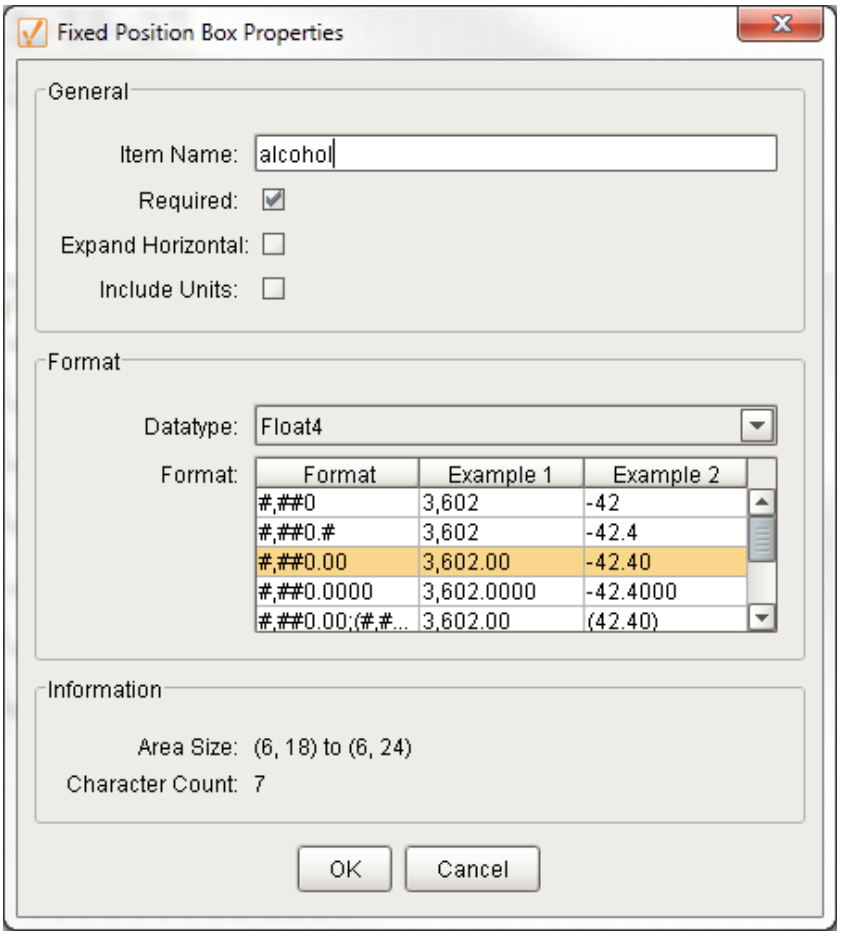
The parsed data will be linked to a label on the template so that the position of the label/value pair can appear at any location.



Find Label Parsing Box Properties

Fixed Position Parsing Box

The parsed data will be read from the template at the exact position the box is placed.

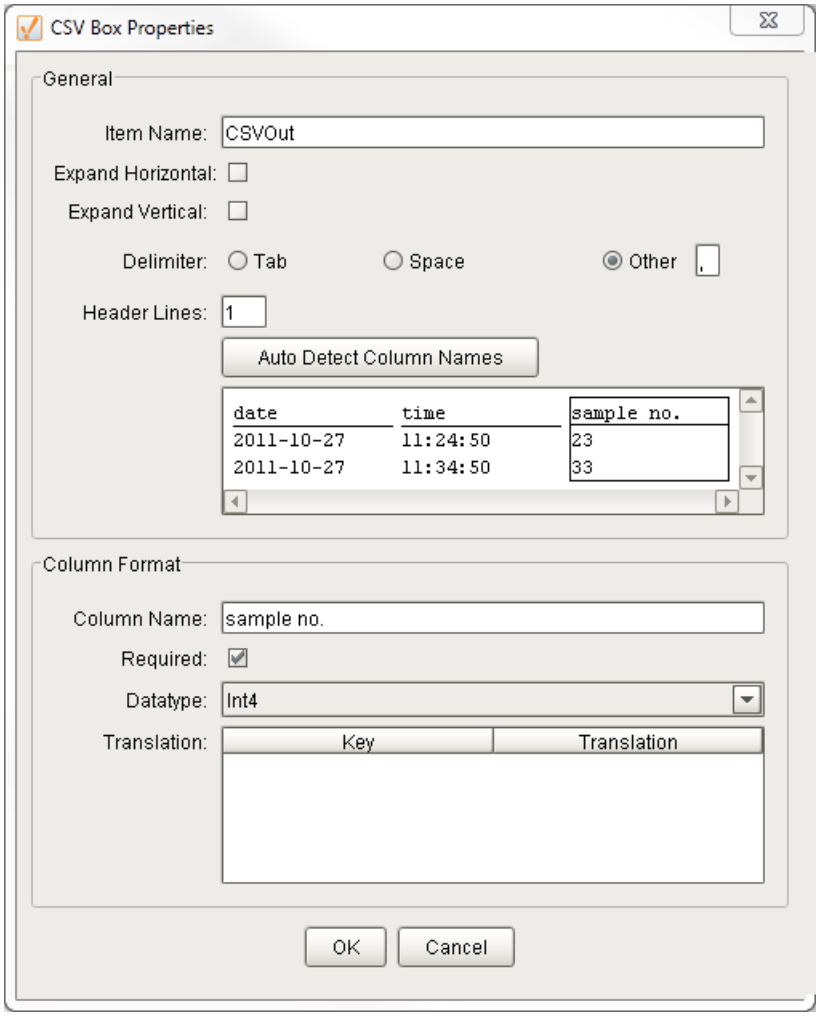


Fixed Position Parsing Box Properties

CSV Column Parsing Box

Will parse all data in the columns in a fashion similar to a CSV file. Rows of data contain repeating items.
For example:

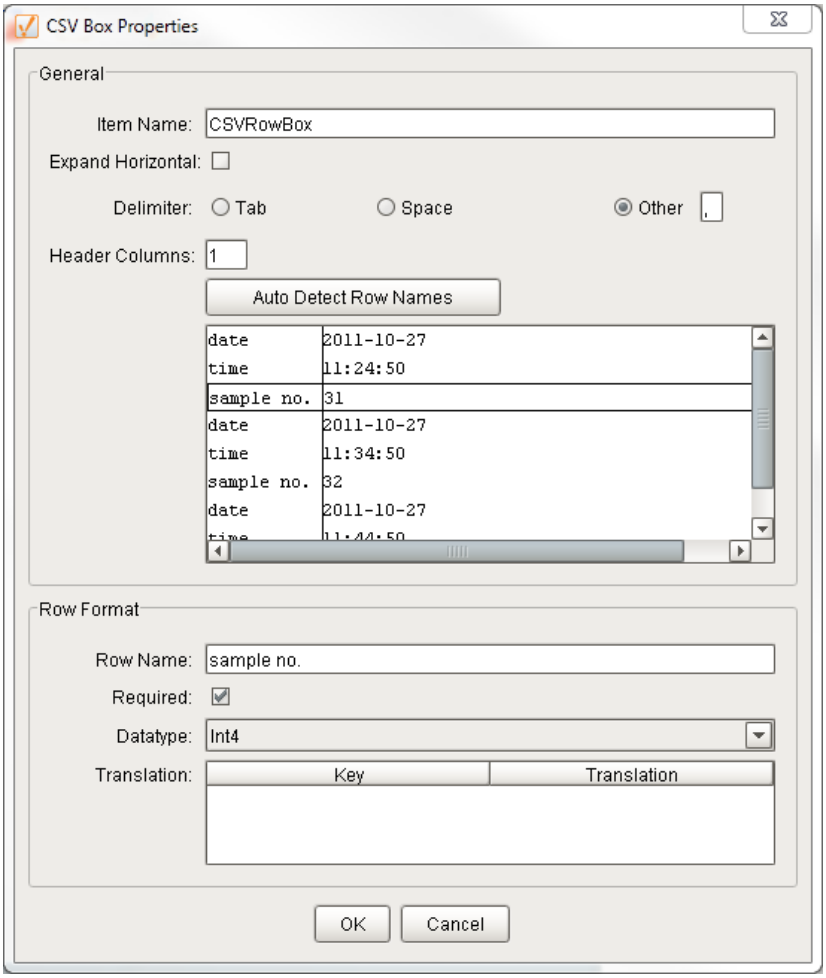
```
date,time,sample no.
2011-10-27,11:24:50,23
2011-10-27,11:34:50,33
```



CSV Row Parsing Box

Will parse all data in the rows in a fashion similar to a CSV file. A group of rows will be repeated.
For example:

```
date,2011-10-27
time,11:24:50
sample no.,31
date,2011-10-27
time,11:34:50
sample no.,32
date,2011-10-27
time,11:44:50
sample no.,33
```



Edit Parsing Box Properties

This will bring up the appropriate editor for the selected parsing box.

Remove Parsing Box

The selected parsing box will be removed.

Toggle Character Grid

A visible grid can be displayed to show the position of all characters.

Parse and Preview Template

This will display a window showing the actual output of the template text after it has been parsed.

5.5 Component Reference

This section is a reference for all of the components that come with the Instrument Interface Module.

5.5.1 File Monitor

Description

An invisible component that handles detecting, reading and parsing functions to provide reading data in files. The term *invisible component* means that this component appears during design time, but is not visible during runtime.

In design time, the last raw data read from a file can be sent to the selected template defined by the Instrument Interface Name by right clicking on the component in the Ignition designer and selecting the Send to Template menu item. This will also select and display the template and replace the existing textual data with the last raw data read.

If the Enable Monitoring property is selected and the designer is preview mode or client has the window open that contains a file monitor component, this component will actively look for files to process. The files that it will process are specified by the File Path property and can contain wildcard characters.

This component will perform a test lock on the file prior to processing to insure that writing to the file is complete. This prevents processing a file before it is ready. This is a important feature If processing of a

file starts and data is still being written to the file it will wither cause errors or incomplete data will be processed.

Properties

This component has standard Ignition properties with the addition of the following properties:

Instrument Interface Name The name of the Instrument Interface configuration to use. The available configurations may be selected by clicking on the pencil icon and selecting from the list or typed in manually.

Scripting name	instrumentInterfaceName
Data Type	String

File Path The file path to monitor for file(s) to process. This can be a path to a single file or it can contain wildcard characters to include multiple files.

By including the "*" or "?" wildcard characters in the file name or even the directory name will switch from single file mode to multi file mode.

The "?" wildcard character represents a single character where the "*" wildcard character represents multiple characters. The sequence "*?" or "?*" is not needed and may not work correctly depending on your operating system.

Examples:

c:\temp\import.csv	Will only process a single file named import.csv
c:\temp*.csv	Will process all files with the csv file extension.
c:\temp\Data*.csv	Will process all files starting with "Data" and have the csv file extension.
c:\temp\Data?.csv	Will process all files starting with "Data" folled by a single character and have the csv file extension. The file name c:\temp\Data10.csv will not be processes but c:\temp\Data1.csv will because the "?" wildcard character is for a single character.
c:\temp**.*	Will process all files contained in directories below the temp directory.

Scripting name	filePath
Data Type	String

Move To Directory Path If the After Processing Handling property is set to "Move File", this is the file path location to move processed files.

Scripting name	moveToDirectoryPath
Data Type	String

After Processing Handling This setting of this property defines how files are handled after processing them.

Scripting name	afterProcessingHandling
Data Type	Integer
Values	Delete File = 0 Move File = 1

File

This property defines the priority to process multiple file. It is inapplicable when a single file is selected in the File Path property. If Alpha Numeric is selected, the files are processed in alphabetical order. If Date is selected, the file names are converted to date values using the pattern defined in the File Name Date Format property and then processed in chronological order. If File Timestamp is selected, the files are processed in chronological order of the file modified date.

Priority

Scripting name	fileProcessingPriority
Data Type	Integer
Values	Alpha Numeric = 0 Date = 1 File Timestamp = 2

File Name Date Format

This property is only applicable if the File Processing Priority property is set to Date. This property defines the parsing pattern to use when converting the file name to a date value when determining the processing order of the files. The patterns can contain both date and time format designators.

Scripting name	fileNameDateFormat
Data Type	String

Example

Pattern	Example File Name
yyyy-MM-dd HH:mm:ss	2012-08-15 13:10:00.csv
yyyy-MMMMM-dd	2012-July-04.txt
MM-dd-yy	10-31-12.log

Letter	Date or Time Component	Example
G	Era designator	AD
y	Year	yyyy = 1996 or yy = 96
M	Month in year	MMMM = July, MMM = Jul, MM = 07 or M = 7
w	Week in year (1-53)	27
W	Week in month (1-5)	2
D	Day in year (1-365)	DDD = 065 or D = 65
d	Day in month	dd = 05 or d = 5
F	Day of week in month	2
E	Day in week	EEEE = Tuesday or EEE = Tue
a	AM / PM marker	AM
H	Hour in day (0-23)	HH = 00 or H = 0
k	Hour in day (1-24)	kk = 08 or k = 8
K	Hour in AM / PM (0-11)	KK = 05 or K = 5
h	Hour in AM / PM (1-12)	hh = 01 or h = 1
m	Minute in hour	mm = 09 or m = 9
s	Second in minute	ss = 01 or s = 1
S	Millisecond	SSS = 890
z	Time zone (general)	zzzz = Pacific Standard Time or zzz = PST
Z	Time zone (RFC 822)	-0800
'	Escape for text	hour' h = hour 9

© Inductive Automation

Enable Monitoring If true automatically detects and processes file(s) contained with the File Path property.

```
Scripting name          enableMonitoring
Data Type               Boolean
```

Monitor Rate The milliseconds between each check for new files. Any files that are found during a check will be processed. Processing of file will not overlap. If the time it takes to process the files exceeds the value of this property, then the next check will be at the next interval.

```
Scripting name          monitorRate
Data Type               Integer
```

Encoding Character encoding.

```
Scripting name          encoding
Data Type               String
```

NOTE: The following properties are not visible in the property editor. They are available for binding and in scripting and expressions.

Last File Processed This property contains the name of the last file processed.

```
Scripting name          lastFileProcessed
Data Type               String
```

Last File Read At The date/time the contents of last file was read.

```
Scripting name          lastFileReadAt
Data Type               DateTime
```

Error Message The current error message or blank if there are no errors.

```
Scripting name          errorMessage
Data Type               String
```

Events

parse - onBeforeParse Is fired before raw data is sent to the parsing engine to be parsed. This provides an method for the raw data to be modified before being parsed. It can be useful to remove unwanted characters or merging more data into the raw data before parsing.

Event Properties

event.getData() Returns the raw data
Data Type String

event.setData(data)

parameters
data — Modified data to send to the parsing engin
Data Type — Data Type String
returns

If data is modified in this event, use this function to write it back to the serial controller component before it is send to the parsing engine.
Data Type String

event.hasData() Returns true if raw data exists.
Data Type Boolean

parse - onAfterParse Is fired after the raw data has been parsed.

Event Properties

event.getParseResults() Returns a ParseResults object containing all the values that were parsed from the raw data. See ParseResults object reference for more information about reading values from the ParseResults object.
Data Type ParseResults

The following script will get results and read a value:

```
results = event.getParseResults()
if results != None:
  if results.isRequiredValid():
    sampleNo = results.getValue("sampleno")
```

event.hasParseResults() Returns true if a ParseResults object exists.
Data Type Boolean

monitorFile - onError Is fired when an error occurs during reading file contents. The errorMessage property can be read to get the error message.

Event Properties

none

Methods

read()

Check existence of and process one files. If multiple files exist only one file is processed because the ParseResults are returned.

parameters (none)

returns Returns a ParseResults object containing all the values that were parsed from the raw data. See ParseResults object reference for more information about reading values from the ParseResults object.
Data Type ParseResults

read(fileName)

Check existence of and process one files. If multiple files exist only one file is processed because the ParseResults are returned.

parameters

`fileName`	File path to file to process if it exists.

Data Type String

returns Returns a ParseResults object containing all the values that were parsed from the raw data. See ParseResults object reference for more information about reading values from the ParseResults object.
Data Type ParseResults

parseText(template, text)

Parses the given text by using the template of templateName

parameters

templateName	The template to use for parsing the text. Data Type String
text	The text to be parsed. Data Type String

returns A ParseResults object (see ParseResults object for information about accessing parsed values contained in the parse results.)
Data Type ParseResults

Sample script for the onAfterParse event. Is demonstrates displaying each parsed value in a label component.

```
parseResults = event.getParseResults()
if parseResults.isValid():
    event.source.parent.getComponent('LabelDate').text = str(parseResults.getValue("Date"))
    event.source.parent.getComponent('LabelTime').text = str(parseResults.getValue("Time"))
    event.source.parent.getComponent('LabelSampleNo').text = str(parseResults.getValue("Sample No"))
    event.source.parent.getComponent('LabelAlcohol').text = str(parseResults.getValue("Alcohol"))
    event.source.parent.getComponent('LabelDensity').text = str(parseResults.getValue("Density"))
    event.source.parent.getComponent('LabelCalories').text = str(parseResults.getValue("Calories"))
```

5.5.2 Serial Controller

Description

An invisible component that handles serial communications and parsing functions to provide instrument device communications. The term *invisible component* means that this component appears during design time, but is not visible during runtime.

In design time, the last raw data received from the communication port can be sent to the selected template defined by the Instrument Interface Name by right clicking on the component in the Ignition designer and selecting the Send to Template menu item. This will also select and display the template and replace the existing textual data with the last raw data received.

In run time, if the Instrument Interface Name property is set, raw data received from the serial communications port will be sent to the parsing engine on the gateway to be parsed. The template used to parse the raw data is named the same as the value of the Instrument Interface Name property.

Properties

This component has standard Ignition properties with the addition of the following properties:

Instrument Interface Name

The name of the Instrument Interface configuration to use. The available configurations may be selected by clicking on the pencil icon and selecting from the list or typed in manually.

```
Scripting name          instrumentInterfaceName
Data Type               String
```

Port

The name of the serial port. The available com ports may be selected by clicking on the pencil icon and selecting from the list or typed in manually.

```
Scripting name          portName
Data Type               String
```

Baud Rate

Serial Communication baud rate.

```
Scripting name          baudRate
Data Type               Integer
Values                  Baud 110 = 0
                        Baud 150 = 1
                        Baud 300 = 2
                        Baud 600 = 3
                        Baud 1200 = 4
                        Baud 2400 = 5
                        Baud 4800 = 6
                        Baud 9600 = 7
                        Baud 19200 = 8
                        Baud 38400 = 9
                        Baud 57600 = 10
                        Baud 115200 = 11
                        Baud 230400 = 12
                        Baud 460800 = 13
                        Baud 921600 = 14
```

Data Bits

Serial communication data bits.

```
Scripting name          dataBits
Data Type               Integer
Values                  DATA BITS 5 = 0
                        DATA BITS 6 = 1
                        DATA BITS 7 = 2
                        DATA BITS 8 = 3
```

© Inductive Automation

Hand Shake

Serial communication flow control methods.

Scripting name	handShake
Data Type	Integer
Values	NONE = 0 CTS DTR = 1 CTS RTS = 2 DSR DTR = 3 XON XOFF = 4

Parity

Serial communication parity.

Scripting name	parity
Data Type	Integer
Values	NONE = 0 EVEN = 1 ODD = 2 MARK = 3 SPACE = 4

Stop Bits

Serial communication number of stop bits.

Scripting name	stopBits
Data Type	Integer
Values	Stop Bits 1 = 0 Stop Bits 2 = 1

Auto Open Port

If true automatically opens the port.

Scripting name	autoOpen
Data Type	Boolean

Clear Buffer Before Send

Clears the receive buffer before sending data.

Scripting name	clearBufferBeforeSend
Data Type	Boolean

Correct CRLF

Corrects any combination of end of line characters to carriage return (CR) and line feed (LF).

Scripting name	correctCRLF
Data Type	Boolean

Default Read Timeout

The default number of milliseconds to wait while reading data.

Scripting name	defaultReadTimeout
Data Type	Integer

© Inductive Automation

Enable Capture

Write all sent and received data to the capture file path.

```
Scripting name          enableCapture
Data Type               Boolean
```

Capture File Path

The file path on the local computer to create the capture file.

```
Scripting name          captureFilePath
Data Type               String
```

Encoding

Character encoding.

```
Scripting name          encoding
Data Type               String
```

Unsolicited Requests

If true, accepts unsolicited requests from the device.

```
Scripting name          unsolicitedRequests
Data Type               Boolean
```

Enable Polled Requests

If true, requests are made at a fixed interval of the Polling Rate property.

```
Scripting name          enablePolledRequests
Data Type               Boolean
```

Polling Rate

Interval in milliseconds to issue poll requests.

```
Scripting name          pollingRate
Data Type               Integer
```

NOTE: The following properties are not visible in the property editor. They are available for binding and in scripting and expressions.

Serial Module Loaded

If true, the serial module has been installed and is loaded.

```
Scripting name          serialModuleLoaded
Data Type               Boolean
```

Serial Port Open

If true, the serial port is open

The following example is the expression binding on a Label component Text property:

```
if({Root Container.Serial Controller.serialPortOpen}, "Ready",
"Not Ready")
```

```
Scripting name          serialPortOpen
Data Type               Boolean
```

Last Data Sent At

The date/time the the latest data has been sent.

```
Scripting name          lastDataSentAt
Data Type               DateTime
```

Last Data Received At

The date/time the latest data has been received.

```
Scripting name          lastDataReceivedAt
Data Type               DateTime
```

Error Message

The current error message or blank if there are no errors.

```
Scripting name          errorMessage
Data Type               String
```

Events

parse - onBeforeParse

Is fired before raw data is sent to the parsing engine to be parsed. This provides an method for the raw data to be modified before being parsed. It can be useful to remove unwanted characters or merging more data into the raw data before parsing.

Event Properties

event.getData()

Returns the raw data
Data Type String

event.setData(data)

parameters
data — Modified data to send to the parsing engin
Data Type — Data Type String
returns
If data is modified in this event, use this function to write it back to the serial controller component before it is send to the parsing engine.
Data Type String

event.hasData()

Returns true if raw data exists.
Data Type Boolean

parse - onAfterParse

Is fired after the raw data has been parsed.

Event Properties

event.getParseResults()

Returns a ParseResults object containing all the values that were parsed from the raw data. See ParseResults object reference for more information about reading values from the ParseResults object.
Data Type ParseResults

The following script will get results and read a value:

```
results = event.getParseResults()
if results != None:
  if results.isRequiredValid():
    sampleNo = results.getValue("sampleno")
```

event.hasParseResults()

Returns true if a ParseResults object exists.

© Inductive Automation

Data Type Boolean

serialPort - onOpen
Event Properties
Is fired when the serial communication port is opened.

serialPort - onClose
Event Properties
Is fired when the serial communication port is closed.

serialPort - onSend
Event Properties
event.getData()
Is fired when data has been sent to the port.
Returns the data that was sent to the serial communication port.
Data Type char[]

serialPort - onReceive
Event Properties
event.getData()
Is fired when data has been received from the serial communication port.
Returns the dat that was received from the serial communications port.
Data Type String if readString() or readUntil() initiated the read, byte[] if readBytes() initiated the read

serialPort - onPoll
Event Properties
Is fired when the serial communications port has been polled for data.
none

serialPort - onError
Event Properties
Is fired when an error occurs on the the serial communication port. The errorMessage property can be read to get the error message.
none

Methods

openPort()

Attempts to open the port. If an error occurs the errorMessage property will be set and an exception will be thrown.

parameters (none)

returns (nothing)

closePort()

Attempts to close the port. If an error occurs the errorMessage property will be set and an exception will be thrown.

parameters (none)

returns (nothing)

© Inductive Automation

writeString(text)

Write value of the text parameter to the communication port. If an error occurs the errorMessage property will be set and an exception will be thrown.

parameters

`text`	The text to write to the port. `Data Type` `String`

returns (nothing)

writeBytes(data)

Write value of the data parameter to the communication port. If an error occurs the errorMessage property will be set and an exception will be thrown.

parameters

`byte []`	The byte array to write to the port. `Data Type` `byte[]`

returns (nothing)

readString()

Reads and returns string data from the communication port. If an error occurs the errorMessage property will be set and an exception will be thrown. If no data is received within the default timeout setting, then an empty string will be returned.

parameters

`(none)`

returns The data read from the port.
`Data Type` `String`

readString(timeout)

Reads and returns string data from the communication port. If an error occurs the errorMessage property will be set and an exception will be thrown. If no data is received within the value specified in the timeout parameter, then an empty string will be returned.

parameters

`timeout`	The time in milliseconds to wait for a response from the port `Data Type` `Integer`

returns The data read from the port.
`Data Type` `String`

readBytes(count)

Reads and returns byte array data from the communication port. If an error occurs the errorMessage

property will be set and an exception will be thrown. If the number of characters specified by the count parameter are not received within the default timeout setting, then any characters received will be returned.

parameters

`count`	The number of bytes to read from the port. `Data Type` `Integer`

returns The data read from the port.
`Data Type` `byte[ ]`

readBytes(count, timeout)

Reads and returns byte array data from the communication port. If an error occurs the errorMessage property will be set and an exception will be thrown. If the number of characters specified by the count parameter are not received within the value specified in the timeout parameter, then any characters received will be returned.

parameters

`count`	The number of bytes to read from the port. `Data Type` `Integer`
`timeout`	The time in milliseconds to wait for a response from the port `Data Type` `Integer`

returns The data read from the port.
`Data Type` `byte[ ]`

readUntil(delimiter, includeDelimiter)

Reads and returns string data from the communication port up until the character specified by the delimiter parameter. If an error occurs the errorMessage property will be set and an exception will be thrown. If the delimiter character is not received within the default timeout setting, then any characters received will be returned.

parameters

`delimiter`	The delimiter to read until. `Data Type` `Char`
`includeDelimiter`	If true the delimiter will be included in the return value. `Data Type` `Boolean`

returns The data read from the port.
`Data Type` `String`

© Inductive Automation

readUntil(delimiter, includeDelimiter, timeout)

Reads and returns string data from the communication port up until the character specified by the delimiter parameter. If an error occurs the errorMessage property will be set and an exception will be thrown. If the delimiter character is not received within the value specified in the timeout parameter, then any characters received will be returned.

parameters

`delimiter`	The delimiter to read until. `Data Type` `Char`
`includeDelimiter`	If true the delimiter will be included in the return. `Data Type` `Boolean`
`timeout`	The time in milliseconds to wait for a response from the port `Data Type` `Integer`

returns The data read from the port.
`Data Type` `byte[ ]`

clearBuffer()

Clear all data existing in the communication port receive buffer. If an error occurs the errorMessage property will be set and an exception will be thrown.

parameters

`(none)`

returns (nothing)

isSerialSupported()

Determines if the client serial module is loaded and available to be used with this component.

parameters

`(none)`

returns True if serial support is available
`Data Type` `Boolean`

parseText(template, text)

Parses the given text by using the template of templateName

parameters

`templateName`	The template to use for parsing the text. `Data Type` `String`

© Inductive Automation

	text	The text to be parsed. Data Type String
returns		A ParseResults object (see ParseResults object for information about accessing parsed values contained in the parse results.) Data Type ParseResults

Sample script for the onAfterParse event. Line 4 shows how to populate measurement values of the SPC module's Sample Entry component.

```
results = event.getParseResults()
if results != None and results.isRequiredValid():
    reading = results.getValue("reading")
    event.source.parent.getComponent('Numeric Text Field').doubleValue = reading.getValue()
    event.source.parent.getComponent('Sample Entry').populateMeasurement("Viscosity", reading.getValue(),
else:
    system.gui.messageBox("Error reading value from instrument.")
```

5.6 Scripting

This section is a reference for scripting functions provided by the Instrument Interface Module. It also has a reference for any objects that are used by or returned by the scripting functions.

5.6.1 Object Reference

The Instrument Interface Module has a parsing engine that takes raw data received from an instrument and from it, extract the desired values. The extracted values can be used to set tags, populate SPC sample measurement values, populate tables, written to database tables and more. Because the extracted values come in various flavors and have various uses, the paring engine returns the extracted values in a ParseResults object.

This section defines the ParseResults object and how to access the extracted values.

5.6.1.1 Parse Results

A ParseResult object is available from the call to getParseResults() on the Serial Controller component.

properties:

- isValid() - Boolean
  If true indicates that all parse values exist and are valid.

- isRequiredValid() - Boolean
  If true indicates the all required parse values exist and are valid.

- get(parseValueType) - List
  Returns a list ParseValue objects of type specified by the parseValueType parameter.

  Available parseValueType options:

  A single, discreet value.
  `system.instrument.parse.types.SingleValue`

  A collections of ParseRow objects.
  `system.instrument.parse.types.RowCollection`

- getAll() - List
  Returns a list of all ParseValue objects.

- getValue(name) - ParseValue
  Returns a ParseValue object for the parsed value specified by the name parameter. The name must match one of the names assigned to a parsing box defined in the parsing template.

- getRowCollection(name) - ParseRowCollection
  Returns a ParseRowCollection object for the name specified by the name parameter. The name must match one of the names assigned to a parsing box defined in the parsing template.

- createDataset(name) - Dataset
  Returns a Dataset object for the parsed value specified by the name parameter. The name must match one of the names assigned to a parsing box defined in the parsing template. This supports converting a ParseRowCollection that is a result of either a CSV Column Parsing Box or a CSV Row Parsing Box into a Dataset. Dataset can be used to display the data in Table or other components in Ignition.

- createValueMap() - Map of name, value pairs.
  Returns a Map object containing name value pairs for all parsed values. The Map can be sent to the SPC module's Sample Entry component to automate populating sample measurement values from an instrument. The Map can also be accessed using scripting.

- createValueMap() - Map of name, value pairs.
  Returns a Map object containing name value pairs for the parsed value specified by the name parameter. The Map can be sent to the SPC module's Sample Entry component to automate populating sample measurement values from an instrument. The Map can also be accessed using scripting.

5.6.1.2 Parse Value

A ParseValue object is available from the get method of the ParseResults object. Because parse values contain additional information such as units, data type, if it is required, etc, the value is contained in this object. The read the true value form the parse value use the getValue() function.

properties:

- isValid() - Boolean
  If true indicates that this parse values is valid.

- isRequiredValid() - Boolean
  If true indicates this parse value is required and is valid.

- isRequired() - Boolean
  If true indicates this parse value is required.

- getName() - String
  Returns name of this parse value.

- getUnits() - String
  Returns the units extracted during parsing for this parse value. The Include Units option must be selected in the parse box options for the units to be extracted.

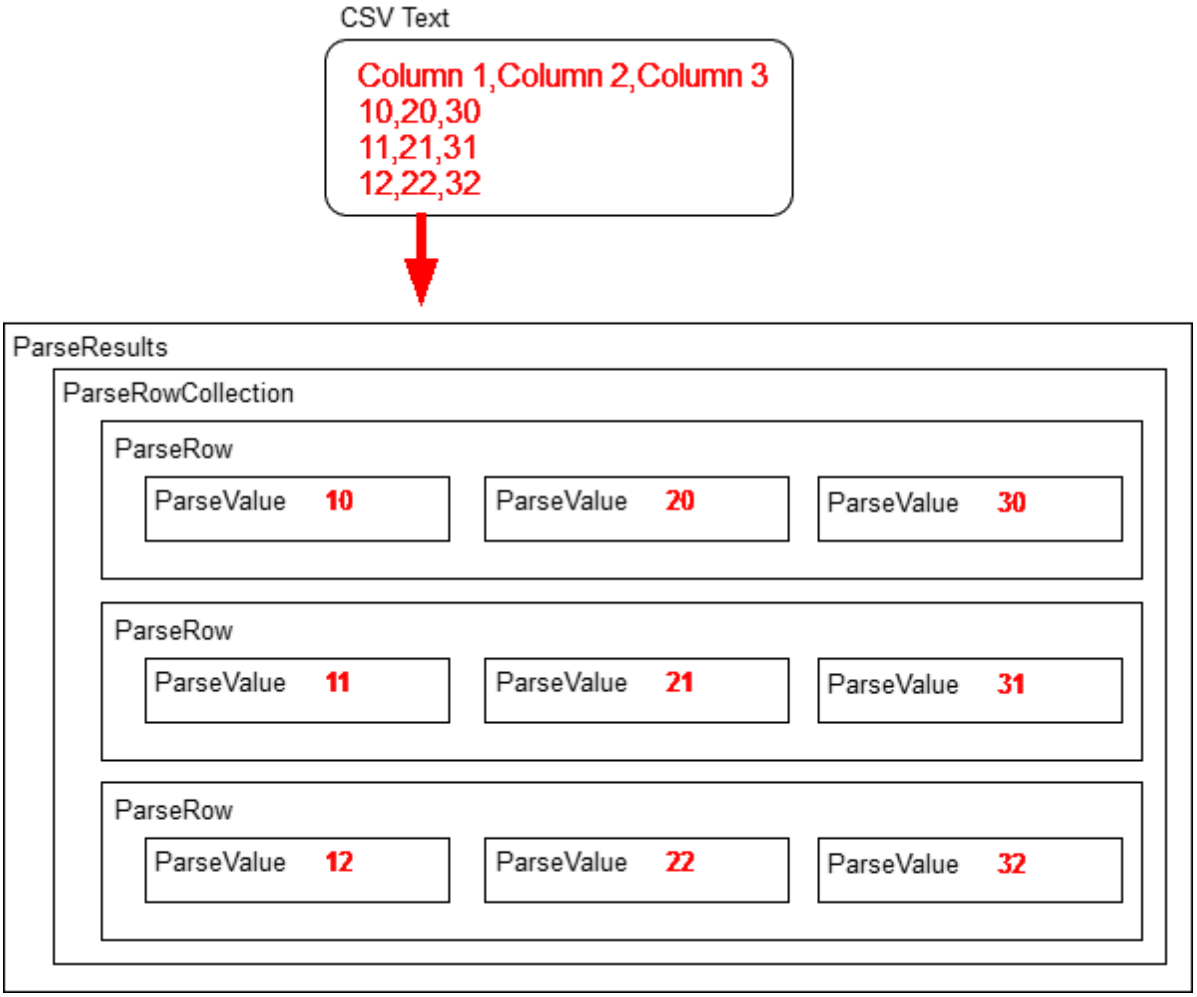
- getDataType() - DataType
  Returns the DataType object of this parse value.

© Inductive Automation

- getValue() - Object
  Returns the true value of this parse value. For example, if the data type defined in the parse box options is a Float8, then a double will be retuned.

5.6.1.3 Parse Row Collection

The Parse Row Collection object contains one or more ParseRow objects. Each ParseRow object contains one or more ParseValue objects. When results contain values from a CSV source, there are rows and columns. As the image below depicts, CSV data is transformed into a ParseResults object.



properties:

- isValid() - Boolean
  If true indicates that all parse values within all parse rows are valid.

- isRequiredValid() - Boolean
  If true indicates that all parse values within all parse rows are required and are valid.

- isRequired() - Boolean
  If true indicates that at least one parse values within all parse rows is required.

- getParseRows() - List of ParseRow objects

© Inductive Automation

Returns a list of all parse rows contained in this collection.

Sample script to cycle though all parse value contained in parse rows:

```
from org.apache.log4j import Logger
log = Logger.getLogger("ParseResult")
fileStr = system.file.readFileAsString("C:\\Temp\\Test.csv")
parseResults = system.instrument.parse.parseText("CSV Test Column", fileStr)
if parseResults.isValid():
    rowCollection = parseResults.getRowCollection("CSV Results")
    parseRowList = rowCollection.getParseRows()
    for parseRow in parseRowList:
           parseValueList = parseRow.getParseValues()
           for parseValue in parseValueList:
                  log.info("%s = %s" % (parseValue.getName(), str(parseValue.getValue())))
```

5.6.1.4 Parse Row

A ParseRow object is available from the getParseRows() funtion of the ParseRowCollection object.

properties:

- isValid() - Boolean
  If true indicates that all parse value objects are valid.
- isRequiredValid() - Boolean
  If true indicates that all parse values objects that are required are valid.
- isRequired() - Boolean
  If true indicates that at least one parse value object is required.
- getParseValues() - List of ParseValue objects
  Results all of the parse values contained in the row.

5.6.2 Gateway Scripts

Methods

```
system.instrument.parse.parseText(projectName, templateName, text)
```

Parses the given text by using the template of templateName

parameters

projectName	The project name where this template is defined.	
	Data Type	String
templateName	The template to use for parsing the text	
	Data Type	String
text	The text to be parsed.	
	Data Type	String

returns

result	Data Type	ParseResults

5.6.3 Client/Designer Scripts

Methods

```
system.instrument.parse.parseText(templateName, text)
```

Parses the given text by using the template of templateName

parameters

templateName	The template to use for parsing the text	
	Data Type	String
text	The text to be parsed.	
	Data Type	String

returns

result	Data Type	ParseResults

Sample script to read and parse a CSV file then convert the parse results to a dataset and display in a table component:

```
fileStr = system.file.readFileAsString("C:\\Temp\\Test.csv")
parseResults = system.instrument.parse.parseText("CSV Test Column", fileStr)
if parseResults.isValid():
    dataset = parseResults.createDataset("CSV Results")
    event.source.parent.getComponent('Table').data = dataset
```

Index

- A -

Adding a Workday Routine Entry 57, 255

- D -

Deleting a Workday Routine Entry 57, 255

- E -

Editing a Workday Routine Entry 57, 255

© Inductive Automation